TRANSPORTATION ANALYSIS SIMULATION SYSTEM (TRANSIMS)

VERSION 1.0

Using TRANSIMS

April 1998

[Tab 2]

Contents

1. TRANSIMS ANALYST INTERFACE (TAI)

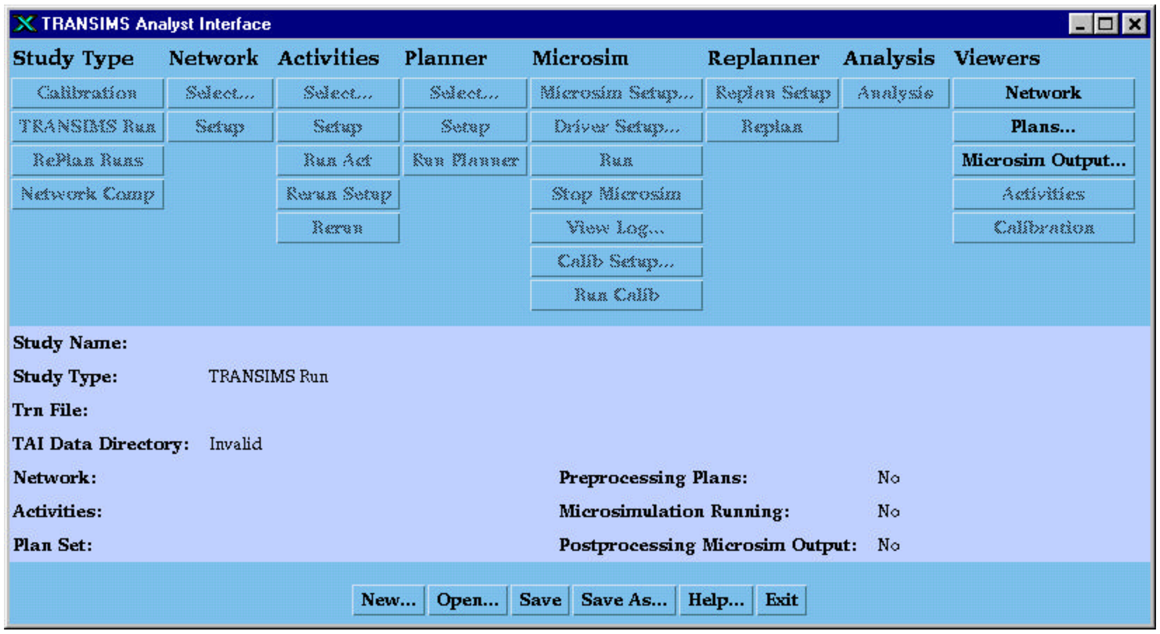
1.1 Initializing the TAI

☛ Assumption: You have installed TRANSIMS properly and set the correct environment variables as described in Tab 1 (*Getting Started*), Section 7 (*Installation Information*).

At the Unix prompt, type

```
tai
```

to start the TRANSIMS Analyst Interface. The TAI Main Screen (Figure 1), showing the initial conditions, is displayed.



Study Name:
Study Type: TRANSIMS Run
Trn File:
TAI Data Directory: Invalid
Network:
Activities:
Plan Set:
Preprocessing Plans: No
Microsimulation Running: No
Postprocessing Microsim Output: No

Figure 1: TAI Main Screen

The basic program buttons are located at the bottom of the TAI main screen. By selecting a basic program button (Figure 2), you may work in one of two modes:

1. Open an *existing* .trn file **[Open…]**

2. Create a *new* study **[New…]**

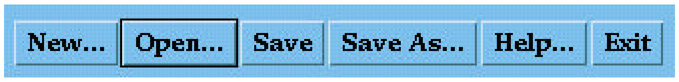


Figure 2: Basic Program Buttons

1.1.1 Opening an Existing .trn File

Click **[Open…]** to display the Open Existing Study window (Figure 3). You may
1) change directories by entering a new directory path in the "Enter path or folder name:" text field,
2) highlight the file name from those shown in the "Files" list box, or
3) type the name of the .trn file you want to open in the "Enter file name:" text field.

You may also change directories by selecting a directory displayed in the "Folders" list, then clicking **[Update]** to display the files in the new directory. Click **[OK]** to choose the file entered in the "Enter file name:" text field.

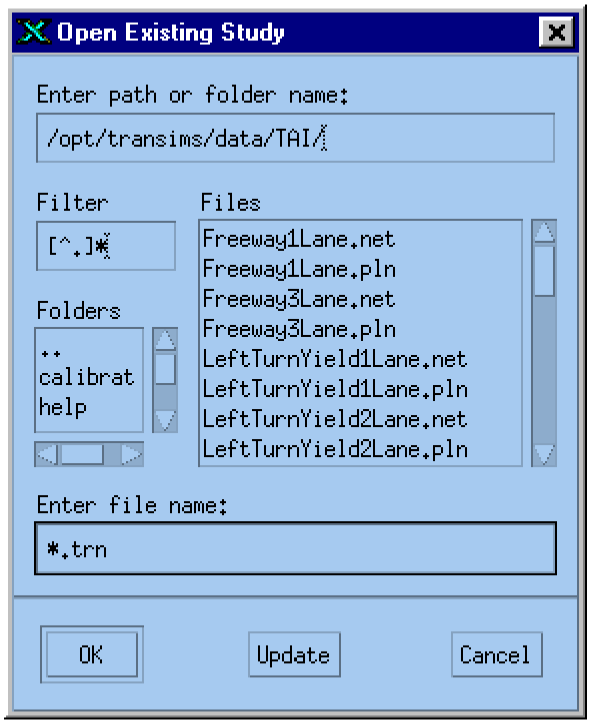


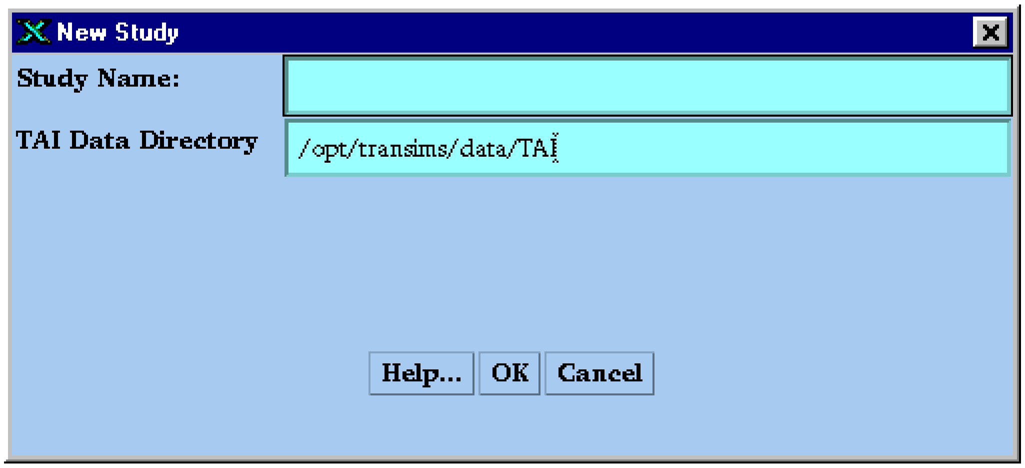
Figure 3: Open Existing Study Window

You may open an existing .trn file by starting up the TAI with the .trn filename as an argument. For example,

```
tai mystudy.trn
```

1.1.2 Creating a New Study

Click **[New…]** to display the New Study window (Figure 4). Type the name of the new study you want to create in the "Study Name:" text field. The "TAI Data Directory" text field will be completed for you with the default environment variable that was set during installation ($TRANSIMS_HOME/data/TAI). Creating a new study resets the TAI to initialization values. Click **[OK].**



Study Name:

TAI Data Directory: /opt/transims/data/TAI

Figure 4: New Study Window

☛ Note: In many windows, you will have the choice of clicking **[OK]** or **[Cancel]**. If you click **[OK]**, the change is applied, but it is not saved to the .trn file until you click **[Save]**. If you click **[Cancel]**, the change is not applied; the window is simply closed.

1.2 TAI Main Screen

The TAI Main Screen (Figure 1) is composed of three areas:

1) The TRANSIMS Components
2) The User Information Area
3) The Basic Program Buttons

1.2.1 TRANSIMS Components

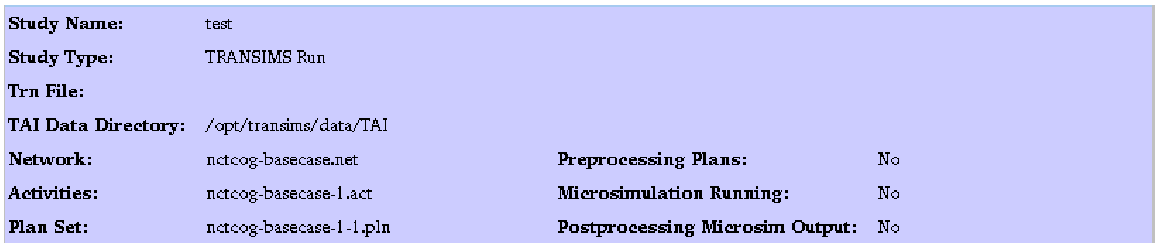
The TAI main screen (Figure 1) provides access to the components of TRANSIMS described in Table 1. These components are displayed as category tabs at the top of the screen.

Table 1: TAI Main Screen Components

Component	Function	See Section
Study Type	Select the mode of analysis (currently Calibration or TRANSIMS Run).	1.2.4.1
Network	Select the transportation network on which you will run the microsimulation. (available only in TRANSIMS Run mode)	1.2.4.2
Activities	Select the set of activities that were generated on a given transportation network. (available only in TRANSIMS Run mode)	1.2.4.3
Planner	Select the set of plans that were generated for a given activity set on a given network. (available only in TRANSIMS Run mode)	1.2.4.4
Microsim	Set up and run the microsimulation to execute the travel plans on a transportation network.	1.2.4.5
Replanner*	Provide feedback between the TRANSIMS Microsimulation and the TRANSIMS Route Planner in which certain travelers change their plans based on the results of the TRANSIMS Microsimulation. *Not implemented in Release 1.0.	1.2.4.6
Analysis*	Provide tools to analyze TRANSIMS results of the Planner, Microsimulation, and Replanner runs. *Not implemented in Release 1.0.	1.2.4.7
Viewers	Provide visualization tools to view TRANSIMS networks, plans, and microsimulation output.	1.2.4.8

1.2.2 User Information Area

User information is displayed in the center of the TAI main screen (Figure 5). User choices are displayed on the left; the status of running programs is displayed on the right.



Study Name:	test		
Study Type:	TRANSIMS Run		
Trn File:			
TAI Data Directory:	/opt/transims/data/TAI		
Network:	nctcog-basecase.net	**Preprocessing Plans:**	No
Activities:	nctcog-basecase-1.act	**Microsimulation Running:**	No
Plan Set:	nctcog-basecase-1-1.pln	**Postprocessing Microsim Output:**	No

Figure 5: User Information

1.2.3 Basic Program Buttons

The basic program buttons (New, Open, Save, Save As, Help, and Exit) are displayed at the bottom of the TAI main screen (Figure 6).

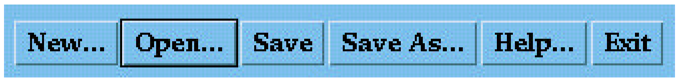
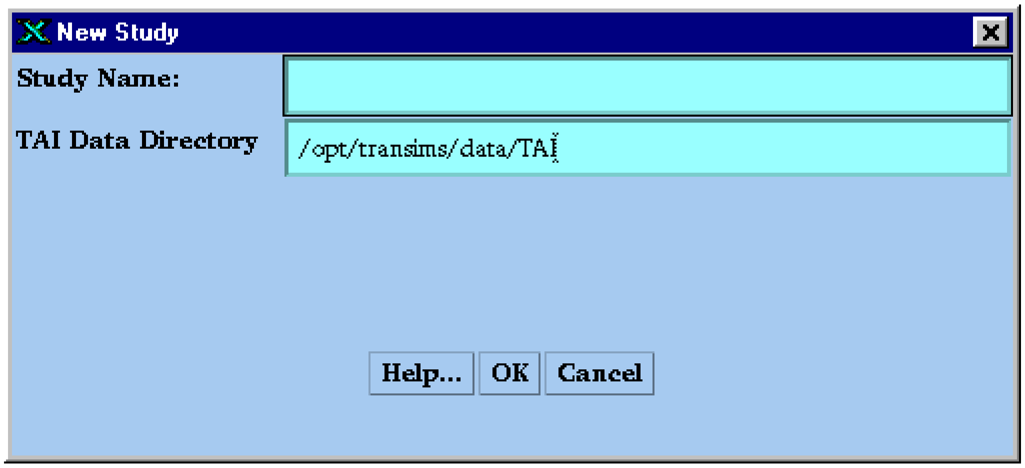


Figure 6: Basic Program Buttons

1.2.3.1 New...

Click **[New…]** to display the New Study window (Figure 7). Type the name of the new study you want to create in the "Study Name:" text field. The "TAI Data Directory" text field will be completed for you with the default environment variable that was set during installation ($TRANSIMS_HOME/data/TAI). Creating a new study resets the TAI to initialization values. Click **[OK]**.



Study Name:

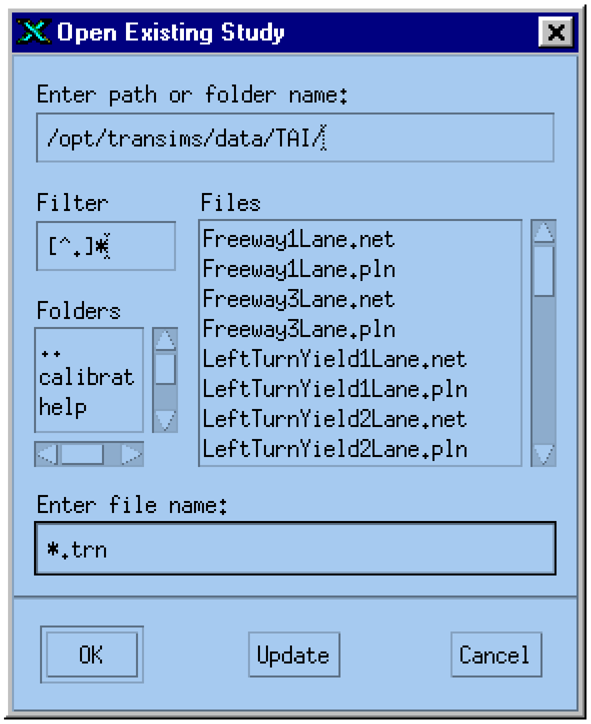
TAI Data Directory /opt/transims/data/TAI

Figure 7: New Study Window

1.2.3.2 Open...

Click **[Open…]** to display the Open Existing Study window (Figure 3). You may
1) change directories by entering a new directory path in the "Enter path or folder name:" text field,
2) highlight the file name from those shown in the "Files" list box, or
3) type the name of the .trn file you want to open in the "Enter file name:" text field.

Click **[Update]** to cause the files displayed in the "Files" list box to be updated and to display the files in the new directory. Click **[OK]** to choose the file entered in the "Enter file name:" text field.



```
/opt/transims/data/TAI/
[^.]*
..
calibrat
help
Freeway1Lane.net
Freeway1Lane.pln
Freeway3Lane.net
Freeway3Lane.pln
LeftTurnYield1Lane.net
LeftTurnYield1Lane.pln
LeftTurnYield2Lane.net
LeftTurnYield2Lane.pln
*.trn
```

Figure 8: Open Existing Study Window

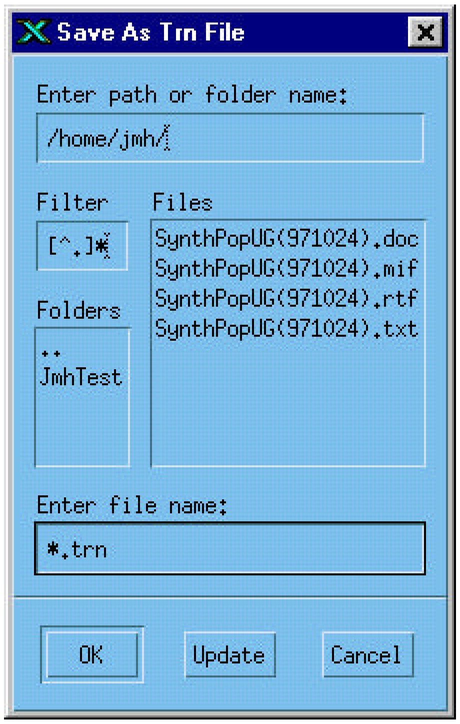
1.2.3.3 Save

Click **[Save]** to save the current interface parameters (user choices) into the current .trn file.

1.2.3.4 Save As...

Click **[Save As...]** to display the Save As Trn File window (Figure 9). You may
1) change directories by entering a new directory path in the "Enter path or folder name:" text field,
2) highlight a file name from those shown in the "Files" list box, or
3) type a new file name in the Enter file name:" text field.

You may also change directories by selecting a directory displayed in the "Folders" list, then clicking **[Update]** to display the files in the new directory. Click **[OK]** to choose the file entered in the "Enter file name:" text field.



/home/jmh/

[^.]*

SynthPopUG(971024).doc
SynthPopUG(971024).mif
SynthPopUG(971024).rtf
SynthPopUG(971024).txt

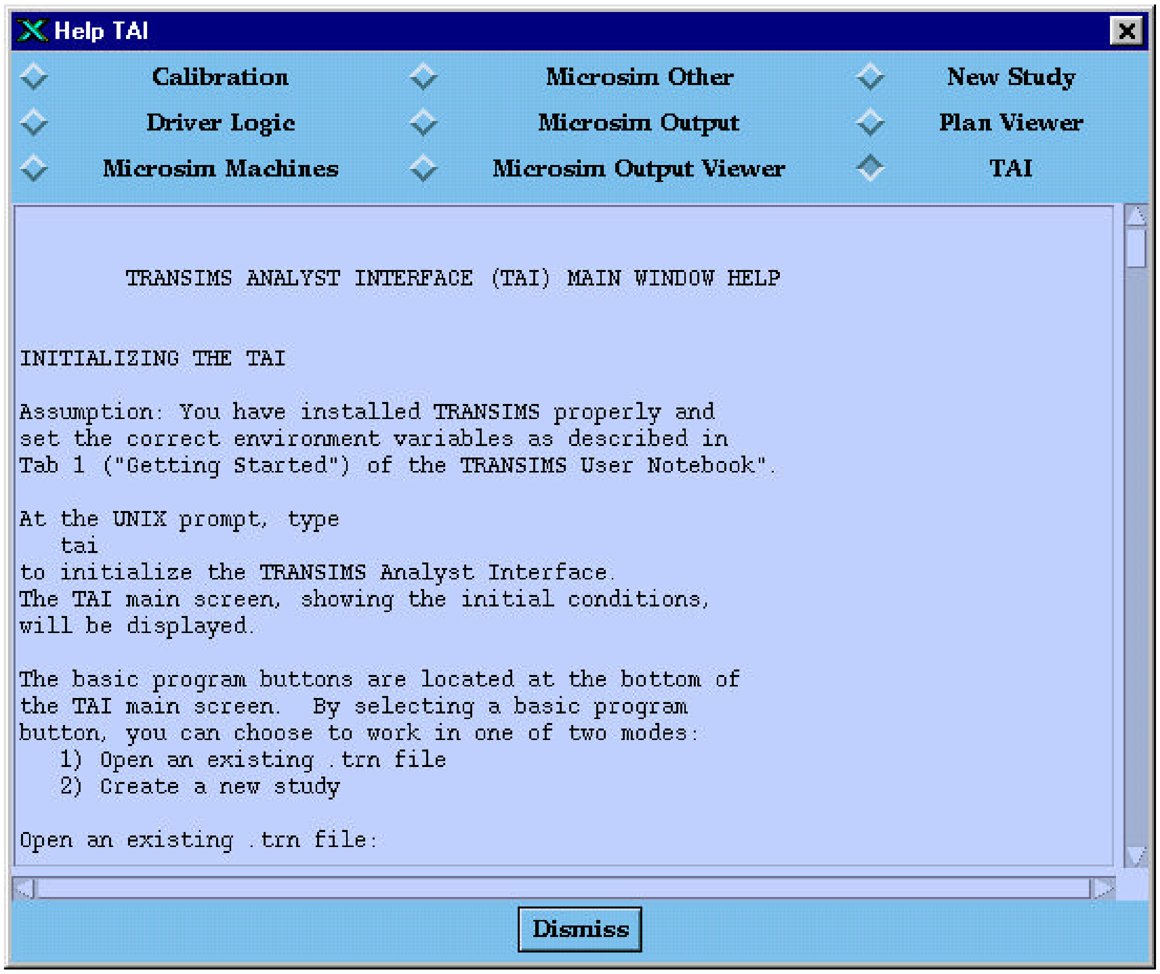
..
JmhTest

*.trn

Figure 9: Save As Trn File Window

1.2.3.5 Help

On-line help is available from several windows, including the Main Screen. Click **[Help…]** to display the "Help" window (Figure 10). Nine Help topics are shown at the top of the Help window. The default Help topic is that of the window you were in when you clicked **[Help…]**. Click another button at the top of the window to display a new Help topic. Click **[Dismiss]** to close the Help window.



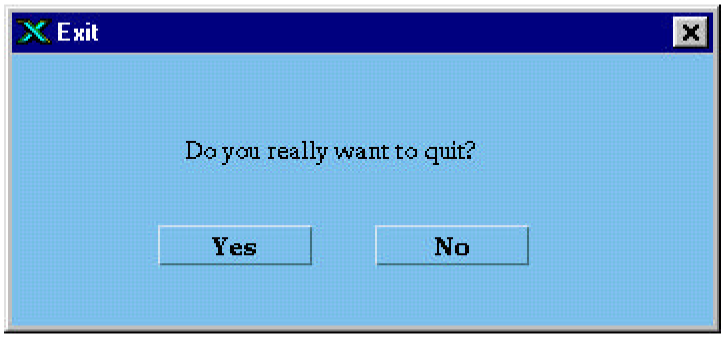
```
        TRANSIMS ANALYST INTERFACE (TAI) MAIN WINDOW HELP


INITIALIZING THE TAI

Assumption: You have installed TRANSIMS properly and
set the correct environment variables as described in
Tab 1 ("Getting Started") of the TRANSIMS User Notebook".

At the UNIX prompt, type
   tai
to initialize the TRANSIMS Analyst Interface.
The TAI main screen, showing the initial conditions,
will be displayed.

The basic program buttons are located at the bottom of
the TAI main screen.  By selecting a basic program
button, you can choose to work in one of two modes:
   1) Open an existing .trn file
   2) Create a new study

Open an existing .trn file:
```

Figure 10: Help Window

1.2.3.6 Exit

Click **[Exit]** to close the program (Figure 11).

☛ There is no reminder to save your changes. This is strictly for exiting!



Do you really want to quit?

Figure 11: Exit Window

1.2.4 Category Tabs

1.2.4.1 Study Type Category

The Study Type category tab contains the function buttons shown in Figure 12. Note that **RePlan Runs** and **Network Comp**arison are not implemented in Release 1.0.

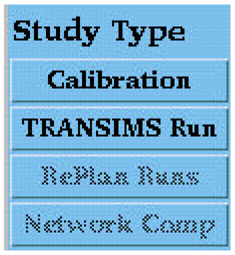
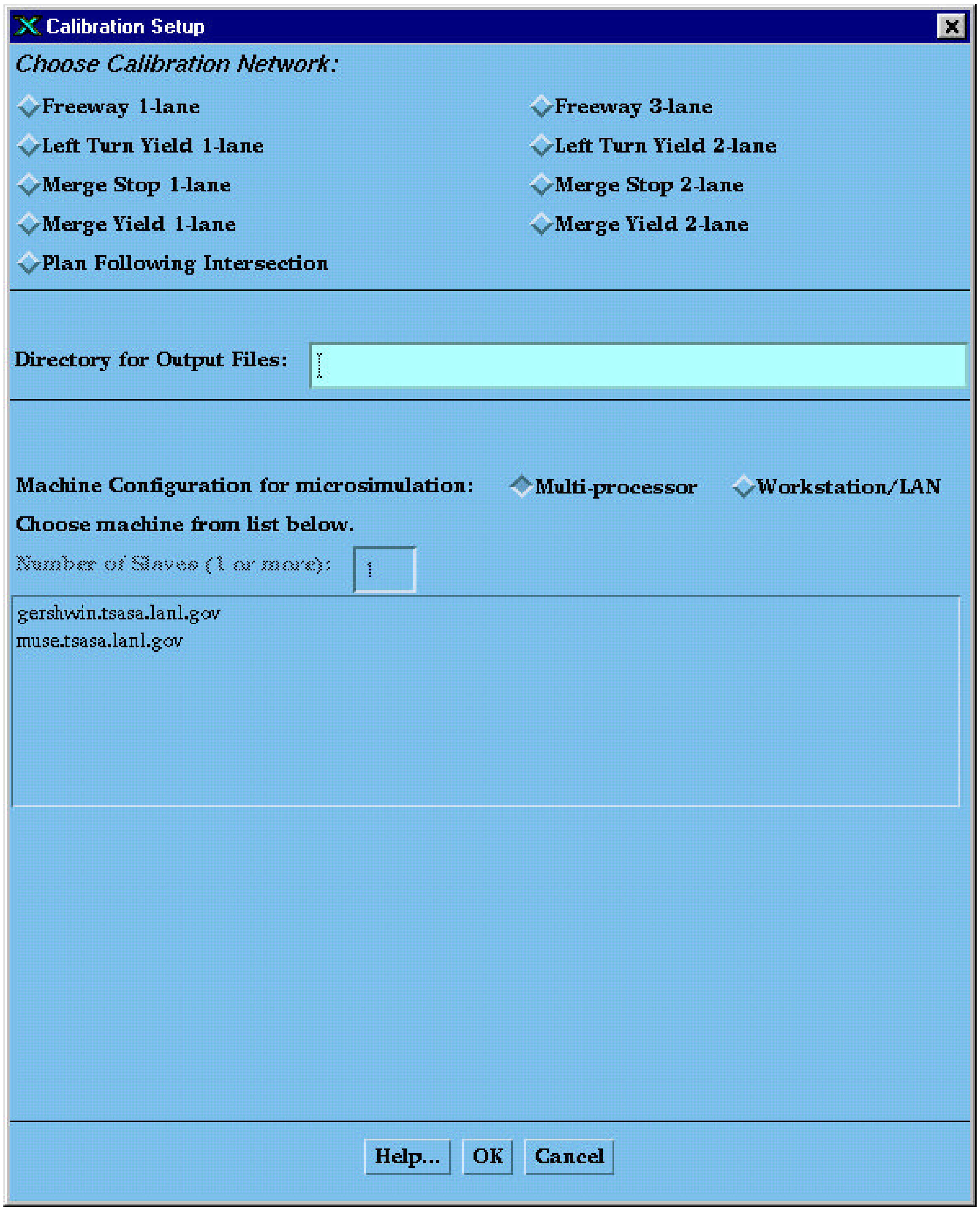


Figure 12: Study Type Category Tab

1.2.4.1.1 Calibration Mode Button

When you select **Study Type→Calibration**, the Calib Setup… function button is enabled. Function buttons under the Network, Activities, and Planner category tabs are disabled (grayed out). Click **[Calib Setup…]** to display the Calibration Setup window (Figure 13). Use Calibration Setup to define the calibration network, the directory for output files, and the machines that will be used for the microsimulation calibration run. Refer to the following sections for more information on setting these parameters. After you have entered the parameters for the calibration run, click **[OK]**. The Calibration Setup window closes, and the Run Calib function button is enabled (see Section 1.2.4.5.7).



Number of Slaves (1 or more): 1

gershwin.tsasa.lanl.gov
muse.tsasa.lanl.gov

Figure 13: Calibration Setup Window

1.2.4.1.1.1 Choose Calibration Network

Calibration Networks are used to examine the traffic dynamics that are produced in the microsimulation. Each network allows you to examine a different dynamic in a controlled experimental network. Table 2 describes the Calibration Networks from which you may choose.

Table 2: Calibration Networks

Calibration Network	Description
Freeway 1-lane	Car following
Freeway 3-lane	Car following and traffic dynamics
Left Turn Yield 1-lane	Left turns across one lane of opposing traffic
Left Turn Yield 2-lane	Left turns across two lanes of opposing traffic
Merge Stop 1-lane	Merging from stop sign into one lane of traffic
Merge Stop 2-lane	Merging from stop sign into two lanes of traffic
Merge Yield 1-lane	Merging from yield sign into one lane of traffic
Merge Yield 2-lane	Merging from yield sign into two lanes of traffic
Plan Following Intersection	Lane changes for plan following during intersection approach

Changes made to the driving logic parameters in the Driver Setup window may produce different traffic dynamics and should be examined by running the microsimulation on the calibration networks.

The simulation start and end times are predefined for each of the calibration microsimulation runs. The calibration networks take from 2 - 12 hours to run. The Plan Following Intersection run is shorter (1 - 2 hours). The Merge and Left Turn calibration runs need at least 250 MB of memory to execute properly.

Microsimulation output collection is predefined for each calibration network. The microsimulation output is postprocessed by a filtering program to condense the collected output into a formatted text file.

1.2.4.1.1.2 Directory For Output Files

Microsimulation output data from the calibration run will be stored in the specified directory.

1.2.4.1.1.3 Machine Configuration for Microsimulation

Two machine configurations are selectable using the Machine Configuration check boxes.

1) Multi-processor. This configuration is used to run the microsimulation processes on a machine with multiple CPUs. Communication between the processes will utilize shared memory. A list of available machines with more than one CPU is displayed. Select a machine from the list. The master and slave processes will both be run on this machine. The number of slaves is always one (1) for calibration runs.

2) Workstation/LAN. This configuration is used to run the microsimulation on single-CPU workstations on a local area network (LAN). Each microsimulation process (master and slave) will execute on a different machine. The LAN is used for communication between the microsimulation processes. Select the machines to be used by the microsimulation during the

calibration run. Two lists of available machines are displayed. Select one machine to serve as the simulation master. Select another machine to serve as the simulation slave. As a machine is selected from either list, it is removed from the other list. If a machine is not listed in the machine lists for Multi-processor or Workstation/LAN but should be, see the Installation section of the Notebook entitled *Setting up for Microsimulation Data Collection*.

1.2.4.1.2 TRANSIMS Run Mode Button

When you select **Study Type→TRANSIMS Run**, the current mode is set to TRANSIMS Run. In TRANSIMS Run mode, you must select (in sequential order) a network, an activity set generated on that network, and a plan set generated on that network activity set. The function buttons **Network→Select**, **Activities→Select**, and **Planner→Select** are enabled/disabled based on your sequential selections. These selections come up automatically where **[New…]** is clicked when creating a New Study.

1.2.4.2 Network Category

The Network category tab contains the function buttons shown in Figure 14. Note that **Setup** is not implemented in Release 1.0.

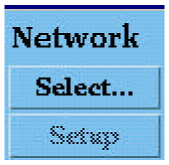


Figure 14: Network Category Tab

1.2.4.2.1 Select... Button

When you select **Network→Select...**, the Select TRANSIMS Network window (Figure 15) is displayed.

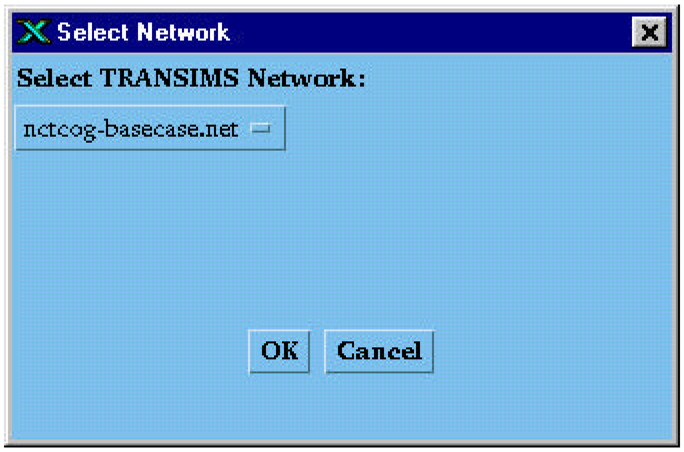


Figure 15: Select TRANSIMS Network Window

Select the TRANSIMS network you want to use from choices in the list box. Click **[OK]**. The current choices are described in Table 3.

Table 3: Available Networks in Release 1.0

Network	Description
nctcog-basecase.net	The base case Dallas/Ft. Worth network as it was in 1990
nctcog-change1.net	An infrastructure change to the base case network with the addition of an extra lane both east and west on the LBJ Freeway
nctcog-change2.net	A set of local changes to the arterial roads of the base case infrastructure

1.2.4.3 Activities Category

The Activities category tab contains the function buttons shown in Figure 16. Note that **Setup**, **Run Act**, **Rerun Setup**, and **Rerun** are not implemented in Release 1.0.

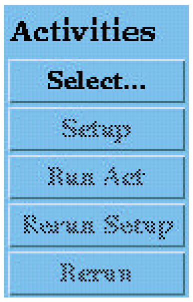
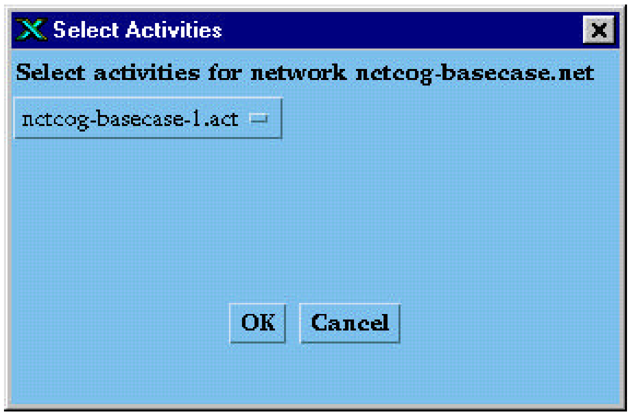


Figure 16: Activities Category Tab

1.2.4.3.1 Select... Button

When you select **Activities→Select...**, the Select Activities window (Figure 17) is displayed.


Select Activities

Select activities for network nctcog-basecase.net

nctcog-basecase-1.act

Figure 17: Select Activities Window

Table 4 shows the available activity sets for each network. At this time, a single set of activities is available for each network. Select an activity set. Click **[OK]**.

Table 4: Networks and Activities

Network	Activities
nctcog-basecase.net	nctcog-basecase-1.act
nctcog-change1.net	nctcog-change1-1.act
nctcog-change2.net	nctcog-change2-1.act

1.2.4.4 Planner Category

The Planner category tab contains the function buttons shown in Figure 18. Note that **Setup** and **Run Planner** are not implemented in Release 1.0.

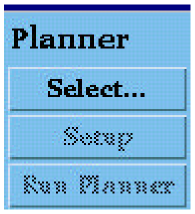
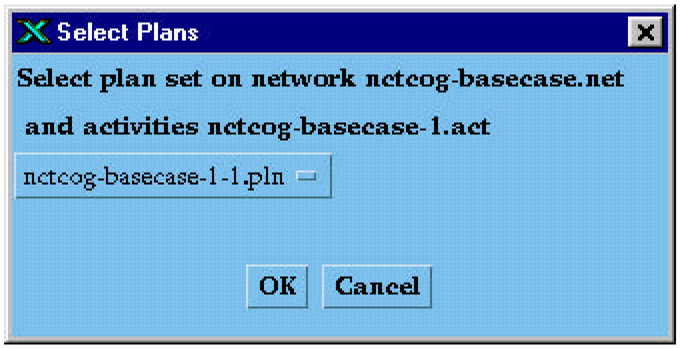


Figure 18: Planner Category Tab

1.2.4.4.1 Select ... Button

When you select **Planner→Select...**, the Select Plan Set window (Figure 19) is displayed.



Select plan set on network nctcog-basecase.net

and activities nctcog-basecase-1.act

nctcog-basecase-1-1.pln

Figure 19: Select Plan Set Window

Table 5 shows the available activity sets and plan sets for each available network. Plan sets were generated on the selected network and based on selected activity sets. In most cases, this release has one plan set per network per activity set. Select a plan. Click **[OK]**.

Table 5: Networks, Activities, and Plans

Network	Activities	Plans
nctcog-basecase.net	nctcog-basecase-1.act	nctcog-basecase-1-1.pln
nctcog-basecase.net	nctcog-basecase-1.act	nctcog-basecase-1-2.pln
nctcog-change1.net	nctcog-change1-1.act	nctcog-change1-1-1.pln
nctcog-change2.net	nctcog-change2-1.act	nctcog-change2-1-1.pln

1.2.4.5 Microsim Category

The Microsim category tab contains the function buttons shown in Figure 20.

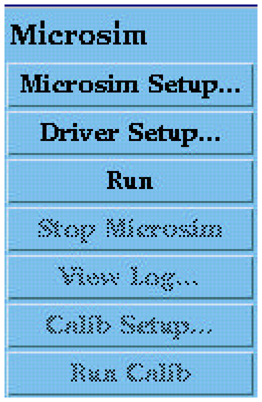
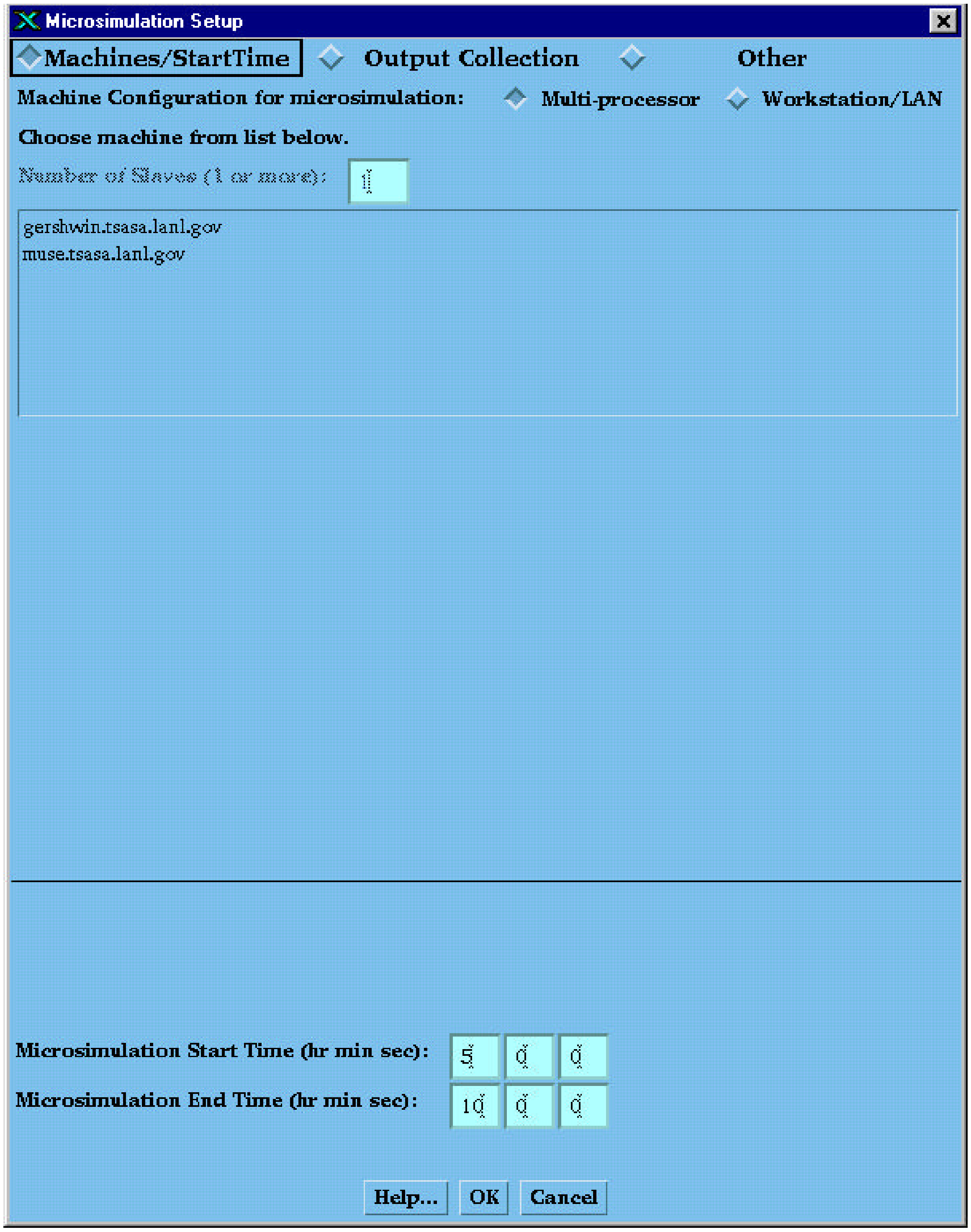


Figure 20: Microsim Category Tab

1.2.4.5.1 Microsim Setup... Button

When you select **Microsim→Microsim Setup...**, (available only in TRANSIMS Run mode) the default Microsimulation Setup **Machine/Start Time** panel (Figure 21) is displayed. Note that you may access the panels (Machines/StartTime, Output Collection, or Other) in any order by using the Machines/StartTime, Output Collection, or Other buttons at the top of the window. Clicking **[OK]** will save your entries on all three panels.



Machine Configuration for microsimulation: Multi-processor / Workstation/LAN

Choose machine from list below.

Number of Slaves (1 or more): 1

gershwin.tsasa.lanl.gov
muse.tsasa.lanl.gov

Microsimulation Start Time (hr min sec): 5 0 0

Microsimulation End Time (hr min sec): 10 0 0

Figure 21: Microsimulation Setup Machine/Start Time Panel

1.2.4.5.1.1 Machines/Start Time

The Machines/StartTime panel is used to select the machines to be used by the simulation and to specify the starting and ending times for the simulation.

1.2.4.5.1.1.1 Machine Configuration for Microsimulation

Two machine configurations are selectable using the Machine Configuration check boxes.

1) Multi-processor. This configuration is used to run the microsimulation processes on a machine with multiple CPUs. Communication between the processes will utilize shared memory. A list of available machines with more than one CPU is displayed. Select a machine from the list. The master and slave processes will be run on this machine. Once the machine is chosen, enter the number of slaves (CPUs) that the microsimulation will use. The maximum number of slaves is one less the number of CPUs on that machine.

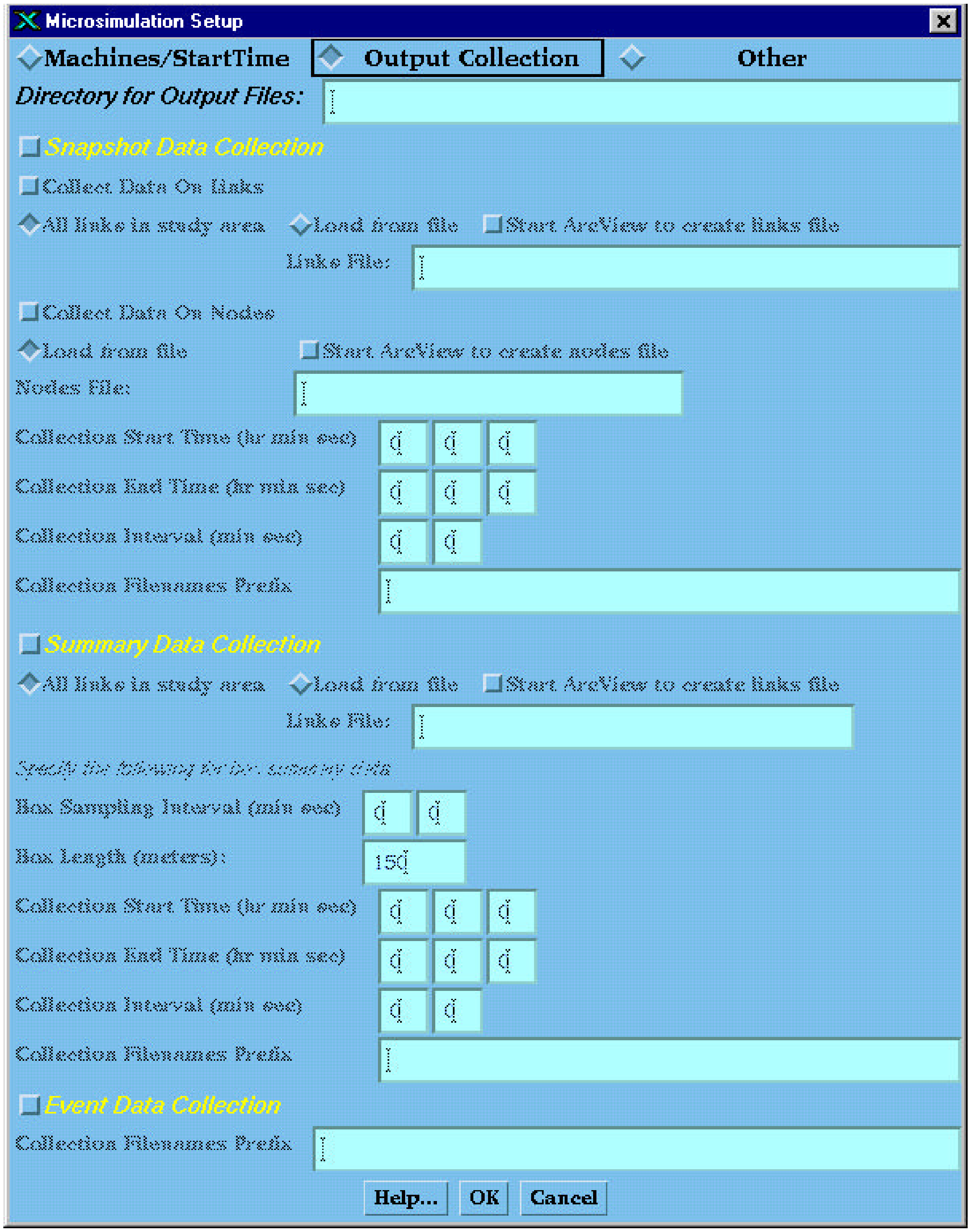
2) Workstation/LAN. This configuration is used to run the microsimulation on single-CPU workstations on a LAN. Each microsimulation process (master and slave) will execute on a different machine. The LAN is used for communication between the microsimulation processes. Select the machines to be used by the microsimulation during the run. Two lists of available machines are displayed. Select one machine to serve as the simulation master. Select machine(s) to serve as simulation slaves. As a machine is selected from either list, it is removed from the other list. If a machine is not listed in the machine lists for Multi-processor or Workstation/LAN but should be, see the Installation section of the Notebook entitled *Setting up for Microsimulation Data Collection*.

1.2.4.5.1.1.2 Microsimulation Start Time / Microsimulation End Time

The microsimulation starting and ending times are specified by entering the hours, minutes, and seconds during a 24-hour period beginning at midnight. Default values are 5:00 a.m. for the starting time, and 10:00 a.m. for the ending time.

1.2.4.5.1.2 Output Collection

When you click **Output Collection** in the Microsimulation Setup window, the Microsimulation Setup Output Collection panel (Figure 22) is displayed.


Microsimulation Setup

Machines/StartTime [✓ Output Collection] Other

Directory for Output Files:

☐ *Snapshot Data Collection*

☐ Collect Data On Links

[✓ All links in study area] Load from file ☐ Start ArcView to create links file

Links File:

☐ Collect Data On Nodes

[✓ Load from file] ☐ Start ArcView to create nodes file

Nodes File:

Collection Start Time (hr min sec)	0	0	0
Collection End Time (hr min sec)	0	0	0
Collection Interval (min sec)	0	0	
Collection Filenames Prefix			

☐ *Summary Data Collection*

[✓ All links in study area] Load from file ☐ Start ArcView to create links file

Links File:

Specify the following for the summary data

Box Sampling Interval (min sec)	0	0	
Box Length (meters):	150		
Collection Start Time (hr min sec)	0	0	0
Collection End Time (hr min sec)	0	0	0
Collection Interval (min sec)	0	0	
Collection Filenames Prefix			

☐ *Event Data Collection*

Collection Filenames Prefix

Help... OK Cancel

Figure 22: Microsimulation Setup Output Collection Panel

The Output Collection panel is used to specify the microsimulation outputs that will be collected while the simulation runs. Three general types of output data are available and independently selectable:

1) Snapshot Data

2) Summary Data

3) Event Data

1.2.4.5.1.2.1 Snapshot Data

Snapshot data is collected when the Snapshot Data Collection box is selected. Snapshot data provides the most detailed information about how the state of the microsimulation evolves in time. Snapshot data may be collected for links and/or nodes. Vehicle data for links consists of the location, velocity, and status of each vehicle; this provides a complete *trajectory* for each vehicle in the simulation. Vehicle data for nodes consists of the location of the vehicle within the intersection buffer. Traffic control data simply reports the current phase and allowed movements at the traffic control. Snapshot data may be collected for each timestep; the data is not summarized (i.e., totaled or averaged) in any way. See Tab 8, *Simulation Output Subsystem* for more details and output formats.

<u>Collect Data On Links</u> When **Collect Data on Links** is selected, you must choose whether to collect the data on all links in the study area, or on a subset of links specified in the Links File. You may enter an existing Links File name or start ArcView to create a new Links File. The Links File contains the ids of the links (one per line). Blank lines are not allowed in this file.

<u>Collect Data On Nodes</u> When **Collect Data On Nodes** is selected, you must specify which nodes to collect data for in the Nodes File. You may enter an existing Nodes File name or start ArcView to create a new Nodes File. The Node File contains the ids of the nodes (one per line). Blank lines are not allowed in this file.

When either links or nodes snapshot data is selected, the collection time parameters must also be specified. Table 6 shows the options and defaults for Snapshot Data.

Table 6: Snapshot Data Options

Option	Description	Default
Collection Start Time (hr min sec)	The starting time for collection of snapshot data, specified in hours, minutes, and seconds during a 24-hour period beginning at midnight.	0 hr, 0 min, 0 sec
Collection End Time (hr min sec)	The time at which snapshot data collection will cease, specified in hours, minutes, and seconds from midnight.	0 hr, 0 min, 0 sec
Collection Interval (min sec)	The frequency at which snapshot data will be recorded. The value is specified in minutes and seconds. A value of 1 second will record information at every timestep and is the most comprehensive type of data collection available.	0 min, 0 sec
Collection Filenames Prefix	The basic part of the snapshot data file names. Extensions indicating the type of the data file will be appended to the prefix to provide the complete file name. When a combination of snapshot, summary, and event data is requested, the Collection Filenames Prefix must be unique for each type of data.	none

1.2.4.5.1.2.2 Summary Data

Summary data reports aggregate data about the microsimulation. The link travel time data consists of counts of vehicles exiting links, and means and variances of the vehicle traversal times for those links. Each link is partitioned spatially into *boxes* of a uniform length, and link density data provides counts and mean velocities of vehicles within the boxes along the link. Note, the default box length is 150 meters. If this is changed, the Microsimulation Output Viewer will not display box summary values correctly.

When summary data is collected, you must also choose whether to collect the data on all links in the study area, or on a subset of links specified in the Links File. You may enter an existing Links File name or start ArcView to create a new Links File. The Links File contains the ids of the links (one per line). Blank lines are not allowed in this file.

When summary data is collected, the sampling and collection time parameters must also be specified. See Tab 8, *Simulation Output Subsystem*, for details and output formats. Table 7 shows the options and defaults for Summary Data.

Table 7: Summary Data Options

Option	Description	Default
Box Sampling Interval (min sec)	The frequency with which the summary data will be sampled and accumulated. The sampling interval is typically shorter than the collection interval so that several samples are aggregated. The value is specified in minutes and seconds.	0 min, 0 sec
Box Length (meters)	The length in meters of the boxes spaced along the link.	150 meters
Collection Start Time (hr min sec)	The starting time for collection of summary data, specified in hours, minutes, and seconds during a 24-hour period beginning at midnight.	0 hr, 0 min, 0 sec
Collection End Time (hr min sec)	The time at which summary data collection will cease, specified in hours, minutes, and seconds from midnight.	0 hr, 0 min, 0 sec
Collection Interval (min sec)	The frequency from which the aggregated summary data will be recorded. The value is specified in minutes and seconds.	0 min, 0 sec
Collection Filenames Prefix	The basic part of the summary data file names. Extensions indicating the type of the data file will be appended to the prefix to provide the complete file name. When a combination of snapshot, summary, and event data is requested, the Collection Filenames Prefix must be unique for each type of data.	none

1.2.4.5.1.2.3 Event Data

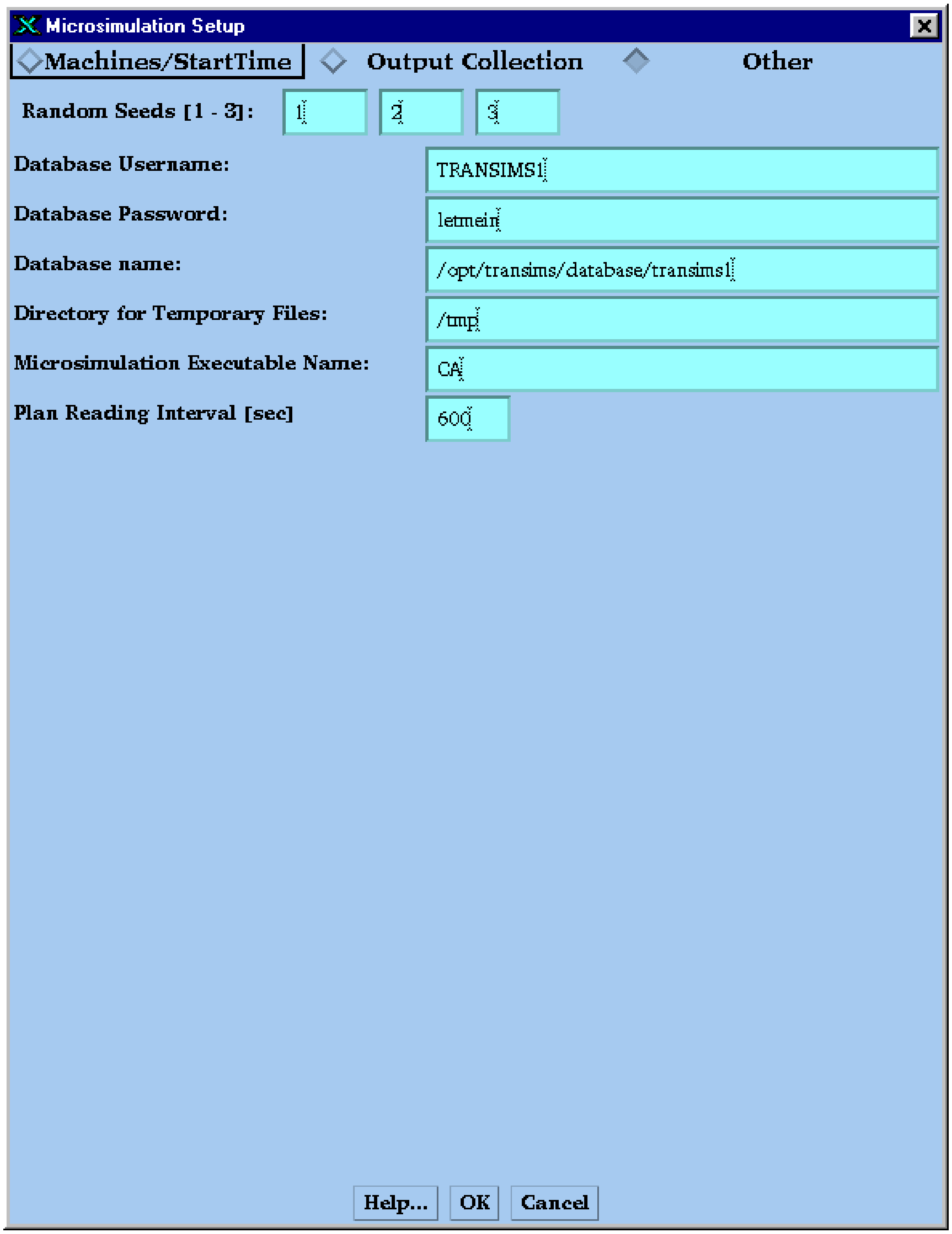
Event Data supplies information on exceptional conditions of vehicles. Examples include when a vehicle becomes *off-plan* (unable to follow its plan), when the plan for a vehicle is invalid, and when the vehicle enters or exits the study area. Event data is collected only when an event occurs. See Tab 8, *Simulation Output Subsystem*, for details and output formats. Table 8 shows the options and defaults for Event Data in Release 1.0.

Table 8: Event Data Options

Option	Description	Default
Collection Filenames Prefix	The basic part of the event data file name. An extension indicating the type of the data file will be appended to the prefix to provide the complete file name. When a combination of snapshot, summary, and event data is requested, the Collection Filenames Prefix must be unique for each type of data.	none

1.2.4.5.1.3 Microsimulation Other

When you click **Other** in the Microsimulation Setup window, the Microsimulation Setup Other panel (Figure 23) is displayed. The Other panel contains miscellaneous parameters. Values for these parameters may be entered by you, but this is usually not necessary because default values are established when the software is installed.



Random Seeds [1 - 3]: 1 2 3

Database Username:	TRANSIMS1
Database Password:	letmein
Database name:	/opt/transims/database/transims1
Directory for Temporary Files:	/tmp
Microsimulation Executable Name:	CA
Plan Reading Interval [sec]	600

Figure 23: Microsimulation Setup Other Panel

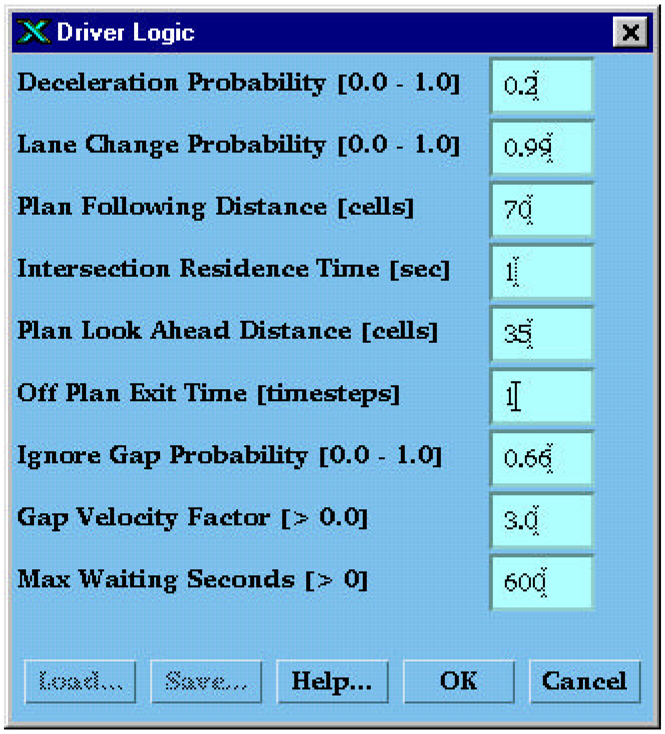
Table 9 shows the options and defaults for Microsimulation Other in Release 1.0.

Table 9: Microsimulation Other

Option	Description	Default
Random Seeds	The random seeds control the initial value of the random number stream used to produce variability in the microsimulation. Valid values for the random seeds are in the range of 1-65535.	1, 2, and 3
Database Username Database Password Database Name	The database parameters control access to the TRANSIMS database tables used by the microsimulation. The Database Name field should specify the directory where the TRANSIMS database has been installed. At some sites, depending on how the database has been configured, the default values may be the only valid values. Default = $TRANSIMS_HOME/database/transims1	(see last line of Description)
Directory for Temporary Files	Various temporary files are created as the microsimulation runs. This directory specifies where the temporary files should be stored.	/tmp
Microsimulation Executable Name	The name of the microsimulation that will be run when the Microsim Run button is pressed. If a complete path name (starting with /) is not given, the user's environment variable PATH will be used to determine the path to the given executable name.	CA
Plan Reading Interval	Specifies how often additional plans will be read into the microsimulation. The default value is to read plans every 600 seconds. A smaller value means fewer plans will be read at one time, while a larger value means more plans will be read less frequently. The value for this parameter affects the size of the memory required by the running microsimulation.	600 seconds

1.2.4.5.2 Driver Setup... Button

When you select **Microsim→Driver Setup...**, the Driver Logic window (Figure 24) is displayed. Note that you can access Help from this window.



Deceleration Probability [0.0 - 1.0]	0.2
Lane Change Probability [0.0 - 1.0]	0.99
Plan Following Distance [cells]	70
Intersection Residence Time [sec]	1
Plan Look Ahead Distance [cells]	35
Off Plan Exit Time [timesteps]	1
Ignore Gap Probability [0.0 - 1.0]	0.66
Gap Velocity Factor [> 0.0]	3.0
Max Waiting Seconds [> 0]	600

Figure 24: Driver Logic Window

The Driver Logic parameters control how drivers and vehicles behave in traffic. Variations in behavior among drivers is accomplished by allowing certain behaviors to vary randomly within limits. Table 10 shows the options and defaults for Driver Logic.

Table 10: Driver Logic

Option	Description	Default
Deceleration Probability [0.0 - 1.0]	Variation in the traffic is enhanced by having each driver randomly decide whether to decelerate for no apparent reason at each timestep. The probability of decelerating is a value in the range 0.0 to 1.0.	0.2
Lane Change Probability [0.0 - 1.0]	Variation in the traffic is reduced by not allowing every driver who would change lanes based on vehicle speed and gaps in the traffic to do so at each timestep. This is done to prevent *lane hopping*. The probability that a driver will change lanes when speed and gaps permit is a value in the range of 0.0 to 1.0.	0.99
Plan Following Distance [cells]	Specifies a count of the number of cells preceding the intersection within which a vehicle will make lane changes to get in an appropriate lane to transition to the next link in its plan. Beyond this distance, lane changing-decisions are based only on vehicle speed and gaps in the traffic. Within this distance, the lane required by the vehicle's plan is also taken into account. As the vehicle nears the intersection, the bias to be in the lane required to stay on plan is increased. Valid values are positive or zero.	70 cells
Intersection Residence Time [sec]	Specifies the number of seconds that a vehicle requires to pass through a signalized intersection. A vehicle resides in an intersection buffer for this amount of time and is then placed on the next link if the first cell on that link is unoccupied. It will remain in the intersection for a longer time if entry to the next link is blocked by another vehicle. Valid values are positive.	1 second

Option	Description	Default
Plan Look Ahead Distance [cells]	The preferred lane for a vehicle to be in as it approaches an intersection depends on the connectivity from the current link to the next link in the plan. In some situations, it is advantageous for the driver to look beyond the next link to subsequent links in the plan when deciding the preferred lane. Plan Look Ahead Distance controls how far ahead the driver will look. A value of 0 indicates that the driver will not look beyond the next link. A positive value indicates that the driver will look at least one additional link beyond the next link in the plan. The number of additional links that will be considered is determined by the lengths of the subsequent links, with link lengths being summed until the accumulated distance is greater than or equal to Plan Look Ahead Distance. Valid values are positive or zero.	35 cells
Off Plan Exit Time [timesteps]	Specifies the number of seconds a vehicle is allowed to deviate from its plan before being removed from the simulation. This prevents off-plan vehicles from wandering on the transportation network. Valid values are positive.	1 timestep
Ignore Gap Probability [0.0 - 1.0]	Drivers at unsignalized intersections wait for a suitable gap in cross traffic before proceeding through the intersection. The deadlock that would occur when vehicles are waiting for each other at multi-way stop/yield signs is prevented by allowing each driver to ignore the gap constraint with some probability. The probability that the drivers at multi-way stop/yield signs will ignore the constraint is a value in the range of 0.0 to 1.0.	0.66

Option	Description	Default
Gap Velocity Factor [>0.0]	At unsignalized intersections and during protected movements at signalized intersections, drivers wait for a suitable gap in cross traffic before proceeding through the intersection. The number of empty cells in a suitable gap is based on the speed of the cross traffic and the gap velocity factor. The gap size is calculated for each lane of the cross traffic. Gap Size = Speed of Oncoming Vehicle * Gap Velocity Factor The gap velocity factor must be greater than 0.0. Note that a vehicle with a speed of 0 results in a suitable gap size of 0, which improves traffic flow in congested conditions.	3.0
Max Waiting Seconds [>0]	Determines the number of seconds that a vehicle will try to enter an intersection. If the vehicle has not moved from the link into or through the intersection in Max Waiting Seconds, the vehicle will abandon its plan and try an alternative movement through the intersection, if one exists. Max Waiting Seconds must be greater than 0, and should be greater than the longest red cycle of the traffic controls in the simulation.	600 seconds

1.2.4.5.3 Run Button

When you select **Microsim→Run** (available only in TRANSIMS mode), the parameters you selected on Microsimulation Setup and Driver Logic are used to start the microsimulation run. The run can have up to three parts:

1) Preprocessing the plans. The microsimulation uses binary plan files that are created by a plan preprocessor from ASCII plan files. If the binary files exist, the plan preprocessor is not run.

2) Running the microsimulation. The microsimulation is run using the parameters specified in the Microsimulation Setup and Driver Logic.

3) Postprocessing the microsimulation output. Postprocessing is done on the microsimulation output files to create ASCII text files that are sorted and collated.

In addition, a window will be opened that displays a log file of these activities. It tells you when the run is done and where the output is stored. The status information display on the main TAI screen changes as the processes are running.

1.2.4.5.4 Stop Microsim Button

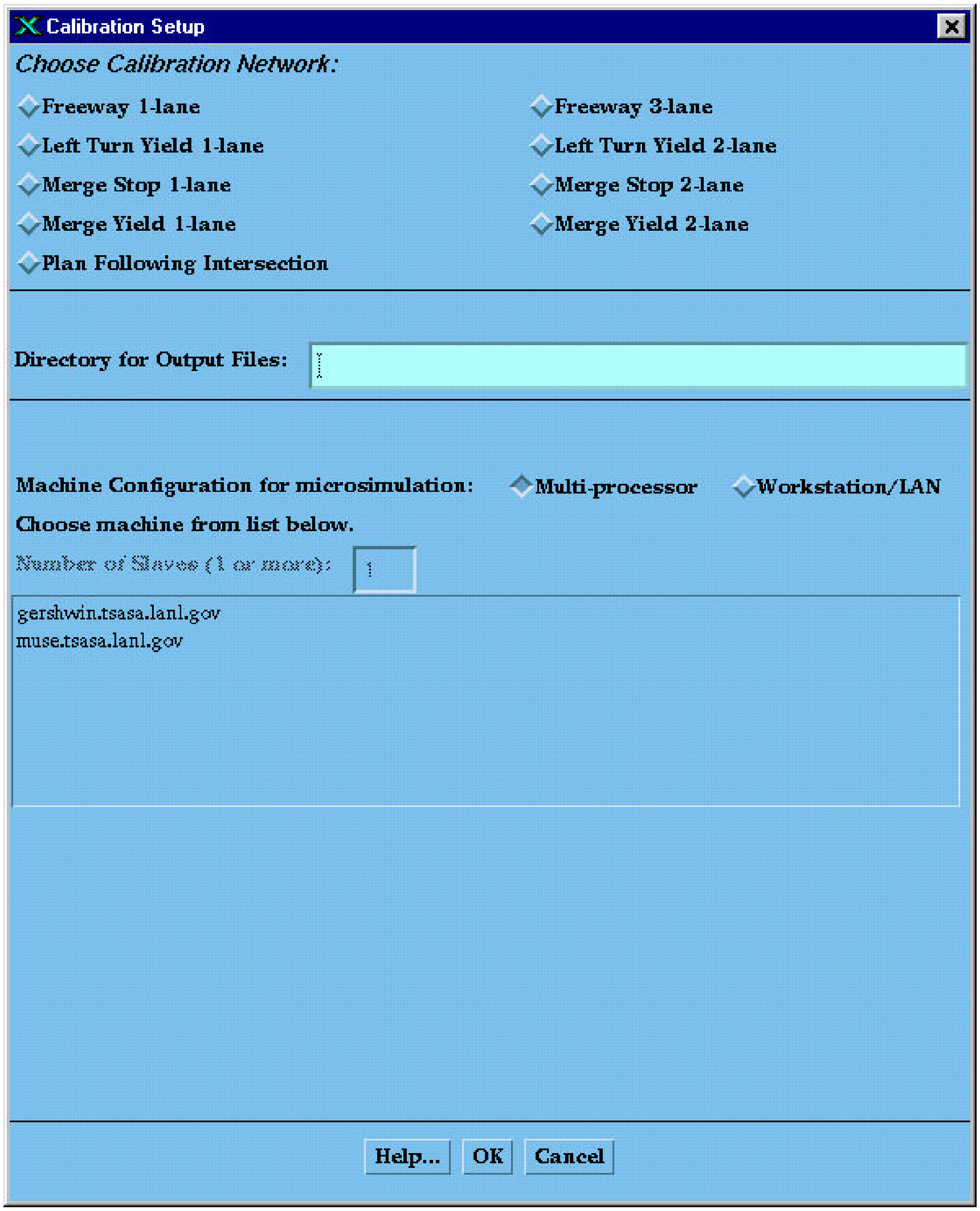
When you select **Microsim→Stop Microsim**, the running processes are killed. You cannot restart from where you stopped the microsimulation.

1.2.4.5.5 View Log... Button

When you select **Microsim→View Log...**, the text of the microsimulation log file from the last microsimulation you ran, or are running, is displayed. Note that the button is enabled only if you have a log file available. If you exit from the TAI when a microsimulation is running, you can use the View Log… button to see the log of the running microsimulation when the TAI is restarted.

1.2.4.5.6 Calib Setup... Button

When you select **Microsim→Calib Setup** (available only in Calibration mode), the Calibration Setup window (Figure 25) is displayed. Refer to Section 1.2.4.1.1 for details.


Calibration Setup

Choose Calibration Network:

Freeway 1-lane

Freeway 3-lane

Left Turn Yield 1-lane

Left Turn Yield 2-lane

Merge Stop 1-lane

Merge Stop 2-lane

Merge Yield 1-lane

Merge Yield 2-lane

Plan Following Intersection

Directory for Output Files:

Machine Configuration for microsimulation: Multi-processor Workstation/LAN

Choose machine from list below.

Number of Slaves (1 or more): 1

gershwin.tsasa.lanl.gov
muse.tsasa.lanl.gov

Figure 25: Calibration Setup Window

1.2.4.5.7 Run Calib Button

When you select **Microsim→Run Calib** (available only in Calibration mode), the calibration is run using the parameters you selected in the Calibration Setup and Driver Logic windows. In addition, a window will be opened that displays a log file of these activities. It tells you when the run is completed and where the output is stored. The status information display on the TAI main screen changes as the parts are running. Postprocessing is done on the calibration output files to create ASCII text files that are sorted and collated. Note: only preprocessed binary calibration plan files are supplied with Release 1.0.

The simulation start and end times are predefined for the calibration microsimulation runs. All of the calibration networks except the *Plan Following Intersection* take 2 - 12 hours to run. The *Plan Following Intersection* run is shorter (1 - 2 hours).

Microsimulation output collection is also predefined for each calibration network.

1.2.4.6 Replanner Category

The Replanner category tab contains the function buttons shown in Figure 26. Note that the Replanner is not implemented in Release 1.0.

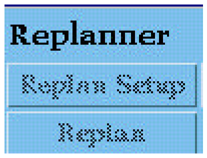


Figure 26: Replanner Category Tab

1.2.4.7 Analysis Category

The Analysis category tab contains the function buttons shown in Figure 27. Note that Analysis is not implemented in Release 1.0.

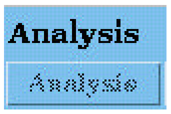


Figure 27: Analysis Category Tab

1.2.4.8 Viewers Category

The Viewers category tab contains the function buttons shown in Figure 28. Three viewers are available: Network, Plans, and Microsim Output. Note that **Activities** and **Calibration** are not implemented in Release 1.0.

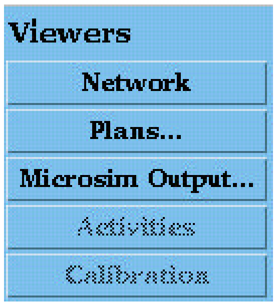


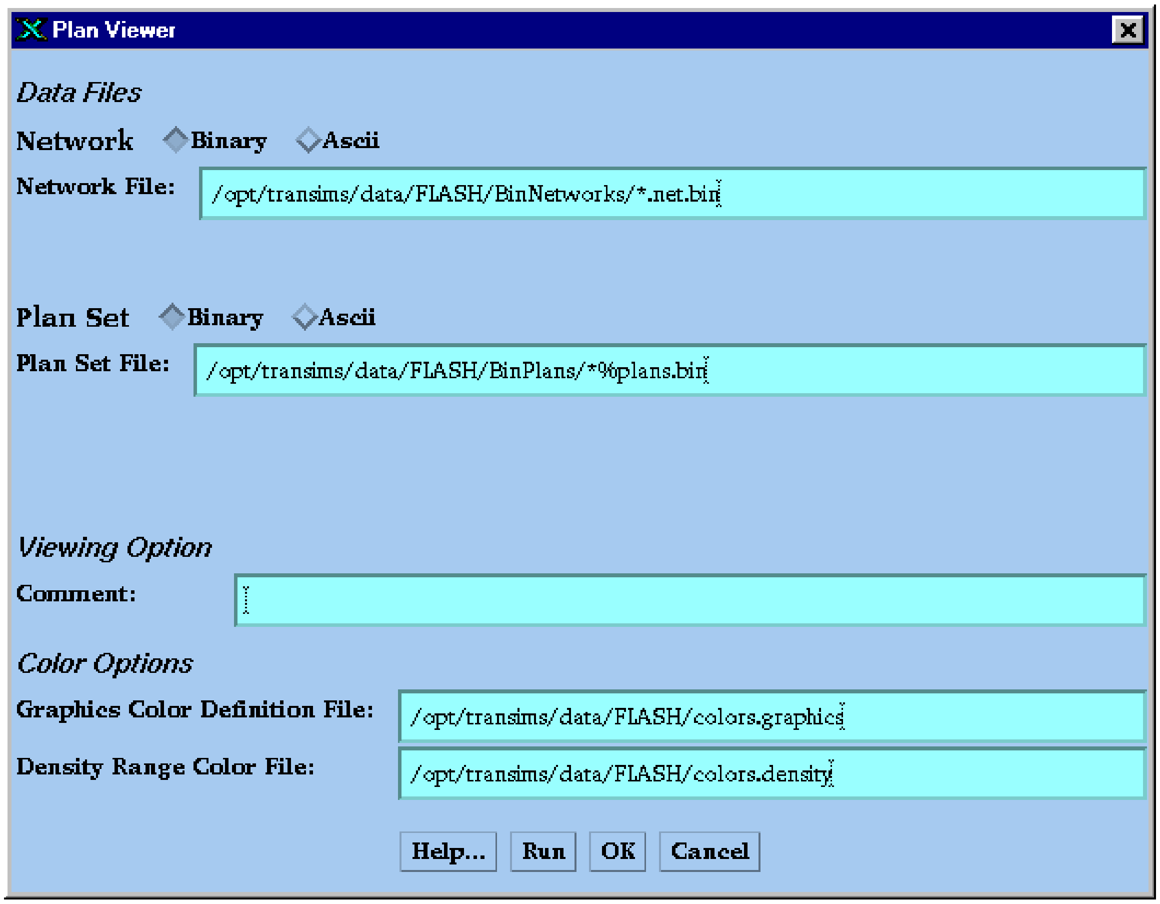
Figure 28: Viewers Category Tab

1.2.4.8.1 Network... Button

When you select **Viewers→Network...**, the TRANSIMS Input Editor, which is an ArcView application, is started. The Input Editor allows viewing of the current transportation network (see Tab 7 for details).

1.2.4.8.2 Plans... Button

When you select **Viewers→Plans...**, the Plan Viewer window (Figure 29) is displayed. See Section 2 for Plan Viewer details.


Plan Viewer

Data Files

Network [✓ Binary] Ascii

Network File: /opt/transims/data/FLASH/BinNetworks/*.net.bin

Plan Set [✓ Binary] Ascii

Plan Set File: /opt/transims/data/FLASH/BinPlans/*%plans.bin

Viewing Option

Comment:

Color Options

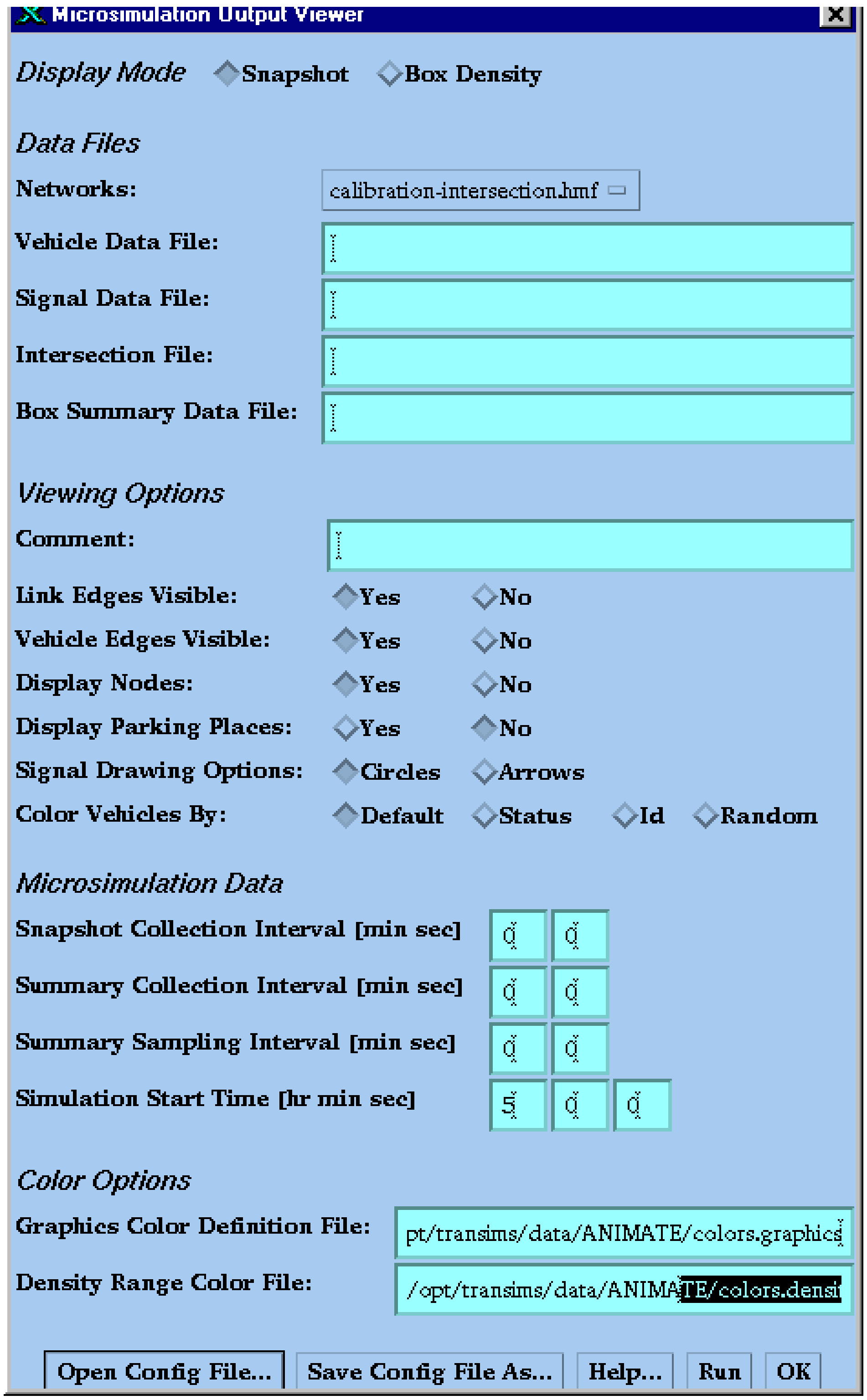
Graphics Color Definition File: /opt/transims/data/FLASH/colors.graphics

Density Range Color File: /opt/transims/data/FLASH/colors.density

Figure 29: Plan Viewer Window

1.2.4.8.3 Microsim Output... Button

When you select **Viewers→Microsim Output...**, the Microsimulation Output Viewer window is displayed (Figure 30). See Section 3 for Microsimulation Output Viewer details.



Microsimulation Output Viewer

Display Mode [✓ Snapshot] Box Density

Data Files

Networks:	calibration-intersection.hmf
Vehicle Data File:	
Signal Data File:	
Intersection File:	
Box Summary Data File:	

Viewing Options

Comment:				
Link Edges Visible:	[✓ Yes]	No		
Vehicle Edges Visible:	[✓ Yes]	No		
Display Nodes:	[✓ Yes]	No		
Display Parking Places:	Yes	[✓ No]		
Signal Drawing Options:	[✓ Circles]	Arrows		
Color Vehicles By:	[✓ Default]	Status	Id	Random

Microsimulation Data

Snapshot Collection Interval [min sec]	0	0	
Summary Collection Interval [min sec]	0	0	
Summary Sampling Interval [min sec]	0	0	
Simulation Start Time [hr min sec]	5	0	0

Color Options

Graphics Color Definition File:	pt/transims/data/ANIMATE/colors.graphics
Density Range Color File:	/opt/transims/data/ANIMATE/colors.densi

Figure 30: Viewers – Microsimulation Output Viewer

2. PLAN VIEWER

2.1 Introduction

The Plan Viewer interface was created for the purpose of visualization and analysis of the ASCII plans created by the Interim Planner. The Plan Viewer was intended to be tightly coupled to the Interim Planner, possibly running Planner algorithms on line. For this reason, it uses Interim Planner output files, not files that have been processed for use by the microsimulator.

2.2 Starting the Plan Viewer

The Plan Viewer is started from the TRANSIMS Analyst Interface (TAI) by running $TRANSIMS_HOME/sunos5.x/bin/tai and then selecting **Plans** under **Viewers**. A setup window is displayed which makes it easier for you to set up and run the Plan Viewer than if you ran it as a standalone. See Section 2.4 for information regarding setting up and running the Plan Viewer from the TAI.

2.3 Setting Up Your Environment

You must set the environment variable DISPLAY to the machine where the Plan Viewer display will appear. To do this, insert a line in your .cshrc file that looks like:

```
setenv DISPLAY    <hostname>:0.0
```

you must also have the location of the Motif library (/usr/dt/lib) and X libraries (/usr/openwin/lib) in the LD_LIBRARY_PATH. For example, to set the Motif library location to /usr/dt/lib:

```
setenv LD_LIBRARY_PATH /usr/dt/lib:${LD_LIBRARY_PATH}
```

2.4 Plan Viewer On-Line Help

2.4.1 Network

The Plan Viewer can be started with either an ASCII or a binary version of the network. Click on the selection button (Binary or ASCII) for the type of network desired. ASCII versions of the network require that you choose a network and a file name for the generated binary version of the network. The node and link tables for the chosen network are read from the database. Binary versions of the network require that you enter the file name of a previously generated binary network. Binary networks are created by the Plan Viewer for optimal access speed. Once the Plan Viewer has run with a particular ASCII network and has created a binary file for that network, subsequent runs on the network should be made using the binary network.

2.4.1.1 Binary Network Specification

Network File – In the binary Network mode, enter the name of the file that contains the binary network that will be used. This file was created by a previous run of the Plan Viewer. Some binary network files have been created for the TRANSIMS Case Study 1 networks. They can be found in the directory <$TRANSIMS_HOME>/data/FLASH/BinNetworks. To get the value of $TRANSIMS_HOME for your installation, type 'printenv TRANSIMS_HOME' at the user prompt in a shell window.

2.4.1.2 ASCII Network Specification

Networks – In the ASCII Network mode, choose the network that the Plan Viewer will use. The network must be the same network used to generate the plan set to be viewed.

Generated Binary Network File – In the ASCII Network mode, enter the name of the file where the binary network will be written.

2.4.2 Plan Set

You have the option of starting up the Plan Viewer with either a binary or an ASCII version of the plans. Click the selection button (Binary or ASCII) to choose the initial plan type. ASCII versions of the plans for TRANSIMS Case Study 1 are included in the TRANSIMS 1.0 release. See your system administrator for the location of these plans. Binary versions of plans are created by the Plan Viewer for optimal access speed. Once the Plan Viewer has run with a particular ASCII plan set and has created a binary file for those plans, subsequent runs on the plan set should be made using the binary plan file.

2.4.2.1 Binary Plan Specification

Plan Set File – In the Binary Plan Set mode, enter the name of the file that contains the binary plan set that was created by a previous run of the Plan Viewer. Some binary plan files have been created for the TRANSIMS Case Study 1 networks. They can be found in the directory <$TRANSIMS_HOME>/data/FLASH/BinPlans.

2.4.2.2 ASCII Plan Specification

Input Plan Set File – In the ASCII Plan Set mode, enter the name of the file that contains an ASCII plan set that was supplied with TRANSIMS 1.0. See your system administrator for the location of these plans.

Sampling Percentage [1.0 - 100.0] – In the ASCII Plan Set mode, enter the percentage of ASCII plans to be converted to binary plans. This is useful when there is a large number of plans in the plan set. Access and update times into the data may be unacceptably slow with increasing numbers of plans. Valid values for this field are in the range from 1 to 100. The default percentage is 5%.

Generated Binary Plan Set File – In the ASCII Plan Set mode, enter the name of the file where the binary Plans will be written.

2.4.3 Viewing Option

Comment – Enter a comment that will appear in the Plan Viewer comment window. This comment is used to differentiate between plan sets.

2.4.4 Color Options

Graphics Color Definition File – Enter the name of a graphics color file. The graphics color file consists of <color use keyword> <color name> pairs for the different types of graphic elements in the viewer. A default graphics color file is provided or one can be created. A default color file is <$TRANSIMS_HOME>/data/FLASH/colors.graphics. If no color file is specified, or if the specified color file cannot be read successfully, default colors will be used for all of the graphic elements. Table 11 lists the recognized color-use keywords. See Section 2.9.3 for a sample graphics color file.

Table 11: Recognized Color-use Keywords

Keyword	Description
text	Color of text
default_background	Window background color when in aggregate or single plans/show Dijkstra mode
single_plan_background	Window background color when in single plans mode for
info_window_background	Legend window background color
text_window_background	Text info window background color
high_res_objects	Color of high resolution objects (squares in middle of links and node circles)
window_outline	Color of all window outlines
default_link	Default link color
single_plan	Color for drawing a single plan
dijkstra_plan	Color for drawing a Dijkstra plan
link_greater_max	Color of links that have greater than the maximum density or flow
solne_grt_max	Color of links whose Dijkstra weight is greater than maximum travel weight so is attached to an unreachable node

Color names can be specified by a valid X11 color name (see /usr/openwin/lib/X11/rgb.txt for valid color names). Or, color names can be specified as a hex RGB value such as #RRGGBB. To use color names that have embedded blanks, you must specify the name with a dash or underscore replacing the spaces. The Plan Viewer translates the dash/underscore in the color name into a space when the color is used. The pound (#) character found at the beginning of a line marks the line as a comment and is ignored by the Plan Viewer.

Example:

```
text                   black

default_background     #808080
```

Density Range Color File – Enter the name of a density color file. The density range color file is used to define the mapping between value ranges and link color when displaying aggregate data. The density color file consists of <density limit> <color name> pairs.

Example density range file:

```
1.0 green
3.0 yellow
6.0 red
```

Color names can be specified by a valid X11 color name (see /usr/openwin/lib/X11/rgb.txt for valid color names). Color names can also be specified as a hex RGB value such as #RRGGBB. To use color names that have embedded blanks, you must specify the name with a dash or underscore replacing the spaces. The Plan Viewer translates the dash/underscore in the color name into a space when the color is used. The pound (#) character found at the beginning of a line marks the line as a comment and is ignored by the Plan Viewer.

The first color specified in the file is for densities less than the first density limit. The second color specified is for densities greater than or equal to the first density limit and for densities less than the second density limit. The last color specified is for densities less than the last density limit. It is assumed that the values are specified in order from lowest to highest. If a density equal to or greater than the last limit occurs, the default color of black is used. A sample density range color file, *colors.density* is in the $TRANSIMS_HOME/data/FLASH directory and can be used as an example density color file. This file should not be changed but can be copied to private space and changed. The private copy can then be specified in this field. See Section 2.9.1 for a sample link density color file.

2.4.5 Help... Button

The Help… button displays a window containing help for the various fields and buttons on the Plan Viewer Setup window.

2.4.6 Run Button

The Run button runs the Plan Viewer. All of the current field values are saved within the TAI, the Plan Viewer Setup window is closed, and the Plan Viewer is started using the configuration file. The field values are also written to a configuration file. A Plan Viewer Output Log window is displayed that contains any output messages from the Plan Viewer.

2.4.7 OK Button

The OK button causes all of the current field values to be saved within the TAI and the Plan Viewer Setup window to be closed.

2.4.8 Cancel Button

The Cancel button closes the Plan Viewer Setup window. If changes were made to any of the fields, those changes will NOT be saved, but the changes will still exist on the window if the Plan Viewer Setup window is brought up again.

2.5 Plan Viewer Display Layout

There are four different areas on the Plan Viewer's display. The menu bar lies across the top of the display and contains menus and menu choices for most of the Plan Viewer's functionality. The large area in the middle of the screen is used to display the network, plans, and aggregate plan data. The bottom of the display is where the comment window is located. It displays the comment string found in the configuration file. The fourth area of the Plan Viewer display is the legend, which is found in the middle of the bottom of the data display area. It displays the current plan file being viewed. For single plan mode, it also displays the number of plans in the binary plan file, the current plan's id, and start link id. For aggregate plans, it displays the time interval and density ranges and their associated colors.

2.6 Plan Viewer Functionality

2.6.1 Menus

The functionality described in Table 12 is accessed via the Plan Viewer menus and menu choices.

Table 12: Plan View Menus

Menu	Command	Description
File		
	Open Binary PlanSet File	Causes a new binary plan file to be read in. The display is updated for the new data. The time interval is reset to the earliest plan time of the new plan file.
	Write Current Link Counts	Causes the link counts to be written to a file. Only the information for the current interval being displayed is written out. The view must be in aggregate mode in order to write out the link information.
	Read Link Counts, Subtract	Used to read in a file of saved link counts that was written by the menu choice above (Write Current Link Counts) and do a subtraction of the saved link counts from the currently displayed link counts. The link colors are then updated to display the differences (what is being differenced depends on the mode the view is in). The view must be in one of the aggregate modes in order to read in the link information. The Plan Viewer will complain if the interval that is currently being displayed does not match the interval that was read in but will still attempt to display the differencing data. Note: differencing HAS to be done on plansets with the same sampling ratios! This differencing capability is good only for the current time interval. Once the timestep is changed, the read in link counts are lost.

Menu	Command	Description
	Print	Used to print a screen dump of the display to a hard copy on a printer. The printer that is used is specified in your PRINTER environment variable. To set this printer variable, add a line to your .cshrc file, then source the .cshrc file.
	Quit	Causes the Plan Viewer to exit.
Find		The Find menu is used to locate either links or nodes on the network displayed. A text input window is displayed. You must click in the entry field in order to give that field focus, then enter in the id of the link or node you wish to locate, then press **[Enter]**. The value entered must be a positive integer. The network view will be shifted in order to center the searched for link or node on the display. A cancel button is available to cancel the find request. If the particular link or node id could not be located within the current network, a message is written to the text information window, which is accessible via the Text menu.
	Link	Uses the functionality listed above to locate a particular link.
	Node	Uses the functionality listed above to locate a particular node.
Mode		The Mode menu is used to control the type of data that is being displayed in the data display area.
	Single Plans	Causes single plans to be displayed on the network. The color of the plan depends on the single_plan color from the graphics color file. In order to move through the plans, see the Frame menu section.
	Single Plans/Show Dijkstra	Causes a single plan and a Dijkstra path for that particular plan’s origination and destination nodes to be displayed on the network at the same time. See the Dijkstra menu section for the different weights used to determine the shortest path.
	Aggregate Plans	Causes the links of the network to be colored by aggregating all of the single plan data for the time interval displayed.

Menu	Command	Description
	Aggregate Plans→Color Map by Flow Demand	Colors the links by flow demand. Counts are counted for those plans that were on a particular link during the time interval, but were not on the link at the end of the interval (hence, the term flow). The final value that is used to color the link is calculated by taking the link counts present in the plan file as described above, multiplying by the sampling ratio and then by the time scale (to make the flow values by hour). Each direction of a link is calculated separately.
	Aggregate Plans→Color Map by Flow Demand/Capacity	Similar to the Color Map by Flow Demand as described above, with the exception of dividing the final count value by the link's capacity. Again, each direction of the link is calculated separately.
View		The View menu is used to control the view of the network.
	Zoom Out	Used to increase the user's view of the network.
	Zoom In	Used to decrease the user's view of the network.
	Reset Map	Used to redraw the network like it was when the Plan Viewer first started up.
	Set to High Res	Used to zoom in continuously until the high resolution mode is reached. The high resolution mode is reached when there are no more than 500 links displayed on the screen. High resolution links have small squares drawn at the center of the link, and circles are drawn for nodes. When the cursor is placed on one of these markers, you may click the left mouse button and information regarding that object is written to the text information window, which is accessible via the Text menu.
	Study Area	Causes the network display to automatically center the Case Study 1 Study Area on the screen.
Utilities		
	Set Time Interval	Used to change the data aggregation interval. The default is 30 minutes. The interval may be anywhere between 1 and 60 minutes.
	Set Time	Used to set the beginning of the time interval to a particular time. A window will come up asking you to enter the hours and minutes in the form of HH:MM. Valid hours entered may be between 00 and 23. Valid minutes may be between 00 and 59.
Dijkstra		Used to change the weight used to determine the Dijkstra path for a particular plan's origination and destination nodes.
	Free Speed Time Weights	Changes the Dijkstra weight to free speeds of the links.

Menu	Command	Description
	Distance Weights	Changes the Dijkstra weight to link distances.
Text		The text information window is used to display information to the user. If you click on a link or node object when you are in high resolution mode, information about that particular object is written to this window. This window also displays warning messages, information about which Dijkstra weight was chosen from the Dijkstra menu, and other information such as what file link count information was written to.
	Show	If the text information window is not currently displayed, choosing this menu item displays the text information window.
	Hide	If the text information window is currently displayed, choosing this menu item closes the text information window.
Frame		In single plans or single plans/show Dijkstra mode, the frame window is used to animate through the plans. • The 'Next' and 'Previous' buttons step through the plans one-by-one under user control. • The 'Reset' button is used to return to the first plan in the plan file. • The 'Start' and 'Stop' buttons start and stop animation through the plans. In aggregate plan mode, the frame window is used to animate through the time intervals. • The 'Next' and 'Previous' buttons step through time intervals one at a time. • The 'Reset' button returns to the first time interval in the plan file. • The 'Start' and 'Stop' buttons start and stop animation through the time intervals.
	Show	If the Frame window is not currently displayed, choosing this menu item displays the Frame window.
	Hide	If the Frame window is currently displayed, choosing this menu item closes the Frame window.

2.6.2 Mouse Operations

Table 13 describes the mouse operations.

Table 13: Mouse Operations

Mouse Action	Function

Mouse Action	Function
Double-click left mouse button	Center on clicked network location
Double-click middle mouse button	Zoom out
Double-click right mouse button	Zoom in
Single-click left mouse button (when in high resolution mode)	Prints information to the text information window on the selected link (little square) or node (little circle)

2.7 Running the Plan Viewer as a Standalone

The Plan Viewer can be run outside the TAI. In order to do this, create a configuration file, then run the Plan Viewer executable $TRANSIMS_HOME/sunos5.x/bin/fp.

2.7.1 Command Line Arguments

The Plan Viewer uses two command line arguments. One allows you to specify the particular configuration file to use. The other allows you to specify the maximum number of plans allowed to be converted to binary.

- **-cf** <configuration file name>   Default: "./pv.in"
- **-nPlans** <#>   The maximum number of plans allowed to be converted to binary

2.7.2 Configuration File

The Plan Viewer uses a configuration file in order to specify data files and viewing options. If you are accessing the Plan Viewer via the TAI, the configuration file will automatically be created by the TAI using the values you set in the **Plans** window under **Viewers**. A sample configuration file can be found in $TRANSIMS_HOME/data/FLASH/pv.in. The configuration file contains four sections:

1) Network specification
2) Planset specification
3) Color specification files
4) Program options

Lines that start with #s within the configuration file are ignored and can serve as comment lines. Input is by keywords and need not be in any particular order.

2.7.2.1 Network Specification

You have the option of starting up the Plan Viewer with either a binary or an ASCII version of the network. ASCII versions of the network consist of a node and a link database table. A run that starts with these tables has the keyword **inputTextNet** included in the configuration file. The table names for the node and link tables are included on lines with the keywords **nodeTableName** and **linkTableName**. The network specified by these tables is cross-correlated (for example, each node has a dynamic array of indices into an array of link structures for its connecting links), and written out as a binary file whose name is keyed with **binNetFileName**. This binary file can be used thereafter to begin runs if the keyword **inputBinNet** is included in the configuration file. Binary versions of the network are created by the Plan Viewer for optimal access speed. Once the Plan Viewer has run with a particular ASCII network and has created a binary file for that network, subsequent runs on the network should be made using the binary network. Table 14 gives possible keywords for network specification.

Table 14: Possible Keywords for Network Specification

Keyword	Description
inputBinNet	Tells the Plan Viewer to use a binary network file
inputTextNet	Tells the Plan Viewer to read in a link and node table then create the binary network file
inputTextNetStop	Tells the Plan Viewer to read in a link and node table, create a binary network file, then exit
nodeTableName <fileName>	Name of table that contains the node information
linkTableName <fileName>	Name of table that contains the link information
binNetFileName <fileName>	Name of the binary network file

Binary network files have been created from the TRANSIMS Case Study 1 networks. They can be found in the $TRANSIMS_HOME/data/FLASH/BinNetworks directory.

2.7.2.2 Planset Specification

You have the option of starting up the Plan Viewer with either a binary or an ASCII version of the plans. ASCII versions of the plans for TRANSIMS Case Study 1 are sent out with the TRANSIMS 1.0 release. See your system administrator for the location of these plans. A run that starts with an ASCII file has the keyword **inputTextPlans** or **inputTextPlansStop** in the configuration file. The former will bring up the Plan Viewer after writing a binary plan file; the latter will write the file and exit. A value must follow both of the above keywords that specifies the sample percentage for writing plans to the binary file. The value may be either an integer or a floating-point number. Converting only a percentage of the plans is useful when there are a large number of plans in the plan set. Access and update times into the data may be unacceptably slow with increasing numbers of plans. Valid values for this field are in the range from 1 to 100. The ASCII plan file is specified with **textPlansFileName**. The binary plan file is specified with **binPlansFileName**. Binary versions of the plans are created by the Plan Viewer for optimal access speed. Subsequent runs may start from the binary plan file if the keyword **inputBinPlans** is specified. Once the Plan Viewer has run with a particular ASCII Plan set and has created a binary file for those plans, subsequent runs on the plan set should be made using the binary plan file. Table 15 gives possible keywords for planset specification.

Table 15: Possible Keywords for Planset Specification

Keyword	Description
inputBinPlans	Tells the Plan Viewer to use a binary plan file
inputTextPlans <sample ratio>	Tells the Plan Viewer to use an ASCII plan file to create a binary plan file
inputTextPlansStop <sample ratio>	Tells the Plan Viewer to use an ASCII plan file to create a binary plan file, then exit
textPlansFileName <fileName>	Name of the file that contains an ASCII planset
binPlansFileName <fileName>	Name of the binary plan file

For example, to create a 5% sample of an ASCII planset:

```
inputTextPlan        20
textPlansFileName    /opt/transims/data/PLANS/Ascii/nctcog-basecase-1-1-plans.txt
binPlansFileName     /opt/TRANSIMS/data/FLASH/BinPlans/nctcog-basecase-1-1-5%plans.bin
```

will write every 20th plan from the ASCII Interim Planner file nctcog-basecase-1-1.txt to the binary file /opt/TRANSIMS/data/FLASH/BinPlans/nctcog-basecase-1-1-5%plans.bin. Note: if you are starting up the Plan Viewer from the TAI, the Plan Viewer setup window will ask for the percentage of plans to convert to binary instead of the ratio.

Some of the information specified in the configuration file is embedded in the binary plan file. The Interim Planner sampling ratio and the Plan Viewer sampling ratio are both saved to the binary file.

Some binary plan files have been created for the TRANSIMS Case Study 1 networks. They can be found in the $TRANSIMS_HOME/data/FLASH/BinPlans directory.

2.7.2.3 Color Specification Files

There are two color specification files. The first is the **graphics color file**. This color file is used to specify the colors of the objects within the Plan Viewer such as the links, nodes, plan paths, etc. The graphics color file consists of <color use keyword> <color name> pairs for the different types of graphic elements in the viewer. A default graphics color file is provided or one can be created. The default color file is $TRANSIMS_HOME/data/FLASH/colors.graphics. If no color file is specified or if the specified color file cannot be read successfully, default colors will be used for the graphic elements. Table 11 shows the recognized color-use keywords.

The second color file is the **link density color file**. This file specifies the mapping between value ranges and link color when displaying aggregate data. The density color file consists of <density limit> <color name> pairs.

Example density range file:

```
1.0 green
3.0 yellow
6.0 red
```

The first color specified in the file will be for values less than the first density limit. The second color specified will be for values greater than or equal to the first density limit and for values less than the second density limit. The last color specified will be for values less than the last density limit. It is assumed that the values will go from lowest to highest within the file. If a value equal to or greater than the last limit occurs, the default color of black will be used. A sample density range color file, *colors.density*, is in the $TRANSIMS_HOME/data/FLASH directory and can be used as an example density color file. This file should not be changed but can be copied to private space and changed. The private copy can then be specified.

For both color files, color names can be specified by a valid X11 color name (see /usr/lib/X11/rgb.txt or /usr/openwin/lib/X11/rgb.txt for valid color names). Or, color names can be specified as a hex RGB value such as #RRGGBB. To use color names that have embedded blanks, you must specify the name with a dash or underscore replacing the spaces. The Plan Viewer will translate the underscore in the color name into a space when the color is used. The pound (#) character found at the beginning of a line marks the line as a comment and is ignored by the Plan Viewer.

The file names of these two color files are specified within the configuration file. To specify these files, you must insert the following lines into the configuration file:

colorFileName	`<path>/`graphics_color_filename
linkDensityColorFileName	`<path>/`link_density_color_filename

2.7.2.4 Program Options

There are two program options. The **plannerSamplingRatio** is the ratio of all plans that are contained in the ASCII planset. Note: the Plan Viewer sampling ratio is specified on the same line as the inputTextPlan keyword. The **comment** keyword is used to specify a comment that will appear in the Plan Viewer comment window. This comment is used to differentiate one set of plans from another.

2.7.3 Specifying the Configuration File

The configuration file can be specified in three possible ways. First, the filename can be specified as a command line argument (see Command Line Arguments above). If a command line argument is not used, the current directory is searched for the file pv.in. If one is not found, default file names are used for the files. The default binary network name is network.bin. The default plan file names are plans.txt and plans.bin. The default configuration is inputBinNet and inputBinPlans. A sample configuration file is in $TRANSIMS_HOME/data/FLASH/pv.in and also appears at the end of this document, along with sample color files.

2.8 Troubleshooting

If unable to allocate enough colors in the default colormap, the PlanViewer creates its own colormap. Switching focus between the PlanViewer and your other displays may be disturbing because of the flashing between colormaps. In order to temporarily fix this, you may type the key sequence <control> <L2> when the desired window has the input focus. The installed colormap will remain installed. In order to switch back to the default colormap, this same sequence will be required when the focus is in another window. Be aware that this affects all future creations of colormaps until your machine is rebooted.

2.9 Sample Files

2.9.1 Sample Link Density Color File

This file is located in $TRANSIMS_HOME/data/FLASH/colors.density. See Section 2.4.4 for information about this file.

```
0.2	#00df00
0.4	#70ef00
0.6	#a0ff00
0.8	#d0ff00
1.0	#ffff00
2.0	#ffd000
3.0	#ffb000
4.0	#ffa000
5.0	#ef7000
6.0	#df0000
```

2.9.2 Sample pv.in File

This file is located in $TRANSIMS_HOME/data/FLASH/pv.in. See Section 2.7.2 for information about this file.

```
######################
### NETWORK mode ###
######################
#inputTextNet
inputBinNet

### Link filename ###
##################
linkTableName		Case Study F Links - Local Streets

### Node filename ###
###################
nodeTableName		Case Study F Nodes - Local Streets

### Binary Network filename ###
############################
binNetFileName		/opt/transims/data/FLASH/BinNetworks/nctcog-basecase-net.bin

#################
### PLAN mode ###
#################
#inputTextPlans  20
inputBinPlans
```

```
### Text Plans filename ###
#######################
textPlansFileName           /opt/transims/data/PLANS/Ascii/nctcog-basecase-1-1-plans.txt

### Binary Plans filename ###
#########################
binPlansFileName            /opt/transims/data/FLASH/BinPlans/nctcog-basecase-5%plans.bin

#################################
### Comment for comment window ###
#################################
comment  Display of Base Case Plans (5% sample)

################
### Color files ###
################
colorFileName                  /opt/transims/data/FLASH/colors.graphics
linkDensityColorFileName   /opt/transims/data/FLASH/colors.density
```

2.9.3 Sample Graphics Color Specification File

This file is located in $TRANSIMS_HOME/data/ FLASH/colors.graphics. See Section 2.4.4 for information about this file.

```
## color of text
text                        black

## default window background color (dark gray)
default_background          #808080

## window background for displaying single plans
single_plan_background      tan

## small info window background
info_window_background      navajo_white

## text info window background
text_window_background      white

## color of high resolution objects (squares in middle of links and node circles)
high_res_objects            black

## color of all window outlines
window_outline              black

## default link color (blue)
default_link                #054094

## color for drawing a single plan
single_plan                 red

## color for drawing a Dijkstra plan
dijkstra_plan               black

## color of links that have greater than the maximum density or flow
link_greater_max            black

## color of links whose Dijkstra weight is greater than maximum travel weight
##   so is attached to an unreachable node.
solne_grt_max               red
```

3. MICROSIMULATION OUTPUT VIEWER

3.1 Introduction

The Microsimulation Output Viewer, also referred to as the animator, is used to visually display the snapshot vehicle, signal, and intersection files, as well as the summary box data file that was produced from a microsimulation run.

3.2 Setting Up the Microsimulation Output Viewer

Animation can be started through the TAI by running $TRANSIMS_HOME/sunos5.x/bin/tai and then selecting **Viewers→Microsim Output**. The Microsimulation Output Viewer window in the TAI makes it easier for a user to set up and run the animator. See Section 3.4, *Microsimulation Output Viewer On-Line Help*, for information regarding setting up and running the Microsimulation Output Viewer from the TAI. The animator can also be run as a standalone tool. See Section 3.9, *Running as a Standalone*, for information about starting up and running the Microsimulation Output Viewer as a standalone.

3.3 Setting Up your Environment

In your .cshrc file, you must specify which display the output will be sent to (the DISPLAY environment variable is not used by HOOPS). To do this, insert a line in your .cshrc file and then source the .cshrc file. Sample line:

```
      setenv HOOPS_PICTURE    X11/<your-hostname>:0.0
```

HOOPS_PICTURE contains the driver to use, X-Windows (X11) in the example above, and the hostname or IP address of the machine where the display will be sent (e.g. foo.lanl.gov: 0.0).

The display can be sent to any machine that has an X server (via appropriate setting of the HOOPS_PICTURE variable).

An OpenWindows server bug may exist that hangs the console when the mouse focus style is not *followmouse*. You should set the following in your .Xdefaults:

```
   OpenWindows.SetInput: followmouse
```

The LD_LIBRARY_PATH environment variable must have the following paths in it: /usr/lib, /usr/openwin/lib, and the location of an xgl library directory, which is typically /opt/SUNWits/Graphics-sw/xgl/lib.

3.4 Microsimulation Output Viewer On-Line Help

3.4.1 Display Mode

The Microsimulation Output Viewer may be started to view **Snapshot** (vehicle, signal, and intersection data if available) or **Box Density** summary data. Click on the selection button to set the initial display mode. The Microsimulation Output Viewer has the capability to switch between the display modes if the data files are specified in the Setup window.

3.4.2 Data Files

Networks – Choose the network for the Microsimulation Output Viewer. This network must be the same as the network used by the microsimulation to create the output data files.

Vehicle Data File – Enter the name of the file that contains the microsimulation output snapshot vehicle data (<filePrefix>.veh.txt).

Signal Data File – Enter the name of the file that contains the microsimulation output snapshot signal data (<filePrefix>.sig.txt).

Intersection File – Enter the name of the file that contains the microsimulation output snapshot intersection data (<filePrefix>.int.txt).

Box Summary Data File – Enter the name of the file that contains the microsimulation output box summary density data (<filePrefix>.box.txt).

3.4.3 Viewing Options

Comment – Enter a comment that will be displayed in the Microsimulation Output Viewer comment window. This comment is used to differentiate one set of data files from another.

Link Edges Visible allows you to specify whether to display link and box edges. More detail is visible when zoomed out if the link edges are not displayed. This option can be changed from within the Microsimulation Output Viewer.

Vehicle Edges Visible allows you to specify whether to display vehicle edges. Vehicle colors are more distinct if the vehicle edges are not displayed. This option can be changed from within the Microsimulation Output Viewer.

Display Nodes allows you to specify whether to display nodes. This option can be changed from within the Microsimulation Output Viewer.

Display Parking Places allows you to specify whether to display parking places. This option can be changed from within the Microsimulation Output Viewer.

Signal Drawing options allow you to specify whether the signals should be drawn as circles or drawn with arrows for protected states.

Color Vehicles By allows you to specify how to color vehicles. Click on the selection button for the method to be used for coloring vehicles. This option can be changed from within the Microsimulation Output Viewer. If **Random** is selected, the vehicles are colored randomly from approximately 13 colors. If **Default** is selected, the vehicles are colored with the color specified in the graphics color definition file. See the section below on the graphics color definition file for more details.

Status Color File – If **Status** is selected as the **Color Vehicle By** option, enter the name of a vehicle status color file. The Vehicle Status Color file defines the mapping between vehicle status and vehicle color. The entries in the file consist of <vehicle status> <color name> <description> triplets. The possible vehicle status values and descriptions are determined by the particular version of the microsimulator being used; however, the vehicle colors maybe modified. The following is an example line from a vehicle status color file:

```
   1   red     is-off-plan
```

If a vehicle's status value is not in the vehicle status color file, the default vehicle color is used. The vehicle status color file that goes with TRANSIMS Release 1.0 is the file $TRANSIMS_HOME/data/ANIMATE/colors.vehstatus. This file should not be modified; however, it can be copied to private space and modified there. Your private copy can then be specified in this field.

Id Color File – If **Id** is selected as the **Color Vehicle By** option, enter the name of a vehicle id color file. The vehicle color id file defines the mapping between vehicle id and vehicle color. The entries in the file consist of <vehicle id> <color name> pairs. The following is an example line from a vehicle color specification file:

```
   1000200   red
```

Those vehicles that are not specified in the file will be colored the default vehicle color. This allows you to specify one less sub-population than is needed within the specification file.

A sample vehicle color specification file, *colors.vehspec*, is in the $TRANSIMS_HOME/data/ANIMATE directory and can to be used as an example color specification file. This file contains the vehicle ids to all of the Galleria travelers from the Case Study 1 Runs and should not be modified; however, it can be copied to private space and modified there. Your private copy can then be specified in this field.

3.4.4 Microsimulation Data

Snapshot Collection Interval [min sec] – Enter the Snapshot Collection Interval used during the microsimulation run that created the data files to be viewed. The time is specified as minutes and seconds and defaults to either a previously entered value or the value of the **Snapshot Data Collection Interval** on the Microsimulation setup window. This value is used to update the clock when displaying snapshot data.

Summary Collection Interval [min sec] – Enter the Summary Collection Interval used during the microsimulation run that created the data files to be reviewed. The time is specified as minutes and seconds and defaults to either a previously entered value or the value of the **Summary Data Collection Interval** on the Microsimulation setup window. This value is used to update the clock when displaying summary data.

Summary Sampling Interval [min sec] – Enter the Summary Box Sampling Interval used during the microsimulation run that created the data files to be viewed. The time is specified as minutes and seconds and defaults to either a previously entered value or the value of the **Box Sampling Interval** on the Microsimulation setup window. This value is used to interpret the density values in the Box Summary data file.

Simulation Start Time [hr min sec] – Enter the Microsimulation Start Time used during the microsimulation run that created the data files to be viewed. It is used to display the animation time. It is specified as hours, minutes, and seconds, and defaults to either a previously entered value or the value of the **Microsimulation Start Time** on the Microsimulation setup window. Time 0 in the data files corresponds to **Simulation Start Time**.

3.4.5 Color Options

Graphics Color Definition File – Enter the name of a graphics color file that contains the color definitions for the graphic objects displayed in the Microsimulation Output Viewer. The entries in the color file consist of <color use keyword> <color name> pairs. Table 16 describes the keywords that are currently recognized.

Table 16: Color Use Keywords

Color Use Keyword	Description
background	Controls the background color of the main graphics drawing window
lane_fill	Controls the fill color of the lane rectangles
lane_outline	Controls the outline color of the lane rectangles
node_marker	Controls the color of the markers placed at the node coordinates
parking_marker	Controls the color of the markers placed at the parking places
vehicle_outline	Controls the outline color or the vehicle geometry
default_vehicle_fill	Controls the fill color of the vehicle geometry when the VEHICLE_COLOR_OPTION Viewing Option is set to COLOR_BY_DEFAULT
highlight_color	Controls the color of objects when they are highlighted
button_outline	Controls the border color of the button windows

A sample color file, $TRANSIMS_HOME/data/ANIMATE/colors.graphics, can to be used as the default color file. This file should not be modified; however, it can be copied to private space and modified there. Your private copy can then be specified in this field.

If no color file is specified or if the specified color file cannot be read successfully, default colors will be used. Table 17 lists the default colors.

Table 17: Default Colors

Color Use Keyword	Default Color
background	blue gray
lane_fill	silver
lane_outline	black
node_marker	silver
parking_marker	yellow
vehicle_outline	black
default_vehicle_fill	blue
highlight_vehicle_fill	yellow
button_outline	blue gray

Density Range Color File – Enter the name of a density range color file. This file defines the mapping between density ranges and link or box colors. It is used when displaying Summary data. The entries in the file consist of <density limit> <color name> pairs. The following is an example link density file:

```
0.1  green
0.3  yellow
1.0  red
```

The first color specified in the file is for densities less than the first density limit. The second color specified is for densities greater than or equal to the first density limit and for densities less than the second density limit. The last color specified is for densities less than the last density limit. It is assumed that the values are specified in order from lowest to highest within the file. If a density equal to or greater than the last limit occurs, the default color of black is used to color the links or boxes. A sample link density color file, *colors.density*, is in the $TRANSIMS_HOME/data/ANIMATE directory and can to be used as the default density color file. This file should not be modified; however, it can be copied to private space and modified there. Your private copy can then be specified in this field.

A Few Notes Regarding File Names – When specifying the path name of each file, do not include the tilde (~) character in the path. The Microsimulation Output Viewer is unable to interpret this character.

A Few Notes Regarding Color Use in Files All color files have several characteristics in common. The pound (#) character found at the beginning of a line marks the line as a comment and is ignored by the Microsimulation Output Viewer. Colors in all four files (vehicle status, vehicle Id, graphics colors, and density range) can be specified by either a HOOPS color string or by a hex RGB value of the format #RRGGBB. HOOPS pre-defines 72 basic color names (see Table 21). To use color names that have embedded blanks, you must specify the name with a dash (-) or underscore (_) character replacing the spaces. The Microsimulation Output Viewer will translate the color string.

3.4.6 Buttons

Open Config File… – Displays a file chooser that allows you to enter the name of a previously created Microsimulation Output Viewer configuration file. The fields in the Microsimulation Output Viewer Setup window are updated with the data contained in the selected configuration file.

Save Config File As ... – Displays a file chooser that allows you to enter the name of a file to save the information on the Microsimulation Output Viewer Setup window. This file can then be accessed at a later time with the 'Open Config File ...' button.

Help... – Displays a window containing help for the various fields and buttons on the Microsimulation Output Viewer Setup window.

Run – Runs the Microsimulation Output Viewer. All of the current field values are saved within the TAI, the values are written to a configuration file, the Microsimulation Output Viewer Setup window is closed, and the Microsimulation Output Viewer is started up using the newly created configuration file. A Microsimulation Output Viewer Output Log window is displayed that contains output messages from the Microsimulation Output Viewer.

OK – Causes all of the current field values to be saved within the TAI and the Microsimulation Output Viewer Setup window to be closed.

Cancel Button – Closes the Microsimulation Output Viewer Setup window. If any changes were made to any of the fields, those changes will NOT be saved, but the changes will still exist on the window if the Microsimulation Output Viewer Setup window is brought up again.

3.5 Microsimulation Output Viewer Display Layout

There are five different areas on the Microsimulation Output Viewer display.

1) The button bar lies across the top of the display and contains buttons for most of the Microsimulation Output Viewer’s functionality.

2) Below the button bar are the comment and time windows. The comment window displays the comment that was present in the configuration file used to start up the Microsimulation Output Viewer.

3) The time window displays the hours, minutes, and seconds of the simulation data being viewed, using SIM_START_TIME and the timestep information in the data files to create this time.

4) The middle area of the screen is the drawing area where the network and data is displayed.

5) Below the middle area is the message window.

6) All status, warning, and error messages will be displayed in the message window.

3.6 Map Manipulation Functionality

There are five buttons that are used to manipulate the view of the network and data. Table 18 describes these buttons.

Table 18: Map Manipulation Functionality Buttons

Button	Description
Zoom In	Decreases your view of the network
Zoom Out	Increases your view of the network
Center	Centers a specific network location on the screen. To use this functionality, click [Center], then click on the network where the new center point will be.
Zoom To	Zooms into a specific area of the network. To do this, click [Zoom To], then click the mouse down in one corner of the area to be zoomed to, slide the mouse (still holding the button down) to the opposite corner of the area to be zoomed to, then release the mouse button. A *rubber band* box will be drawn to show you where the zoom area will be. Do not be alarmed when you use this button for the first time after starting up the Microsimulation Output Viewer. The screen will redraw itself before the zoom area is drawn. After the first Zoom To usage, the display will no longer redraw the screen when using this button.
Reset Map	Redraws the network like it was when the Microsimulation Output Viewer first started up.

3.7 Animate Functionality

There are five buttons that are used to animate through the data files. Table 19 describes these buttons.

Table 19: Animating Functionality Buttons

Button	Description
Step	Causes one timestep to be read from the data file(s). If you are displaying snapshot data, the vehicles on the display are redisplayed in their new position, the signal colors are updated, and the intersection queue information is updated. If summary data is being displayed, the link or box colors change according to their new density values.
Step Back	Causes the timestep previous to the current timestep to be read from the data files, and the display is updated.
Animate	Causes all of the timesteps in the data file(s) to be read, starting at the current time, and the screen to be updated for each timestep read until you click [Stop] or until the data file(s) run out of data.
Reset Time	Causes the first timestep in the data file(s) to be displayed. This button can also be used to set a valid time when the user has switched to a different **Display Mode** and the new data file(s) does not contain data for the current timestep.
Set Time	Used to jump to a particular time in the data files. A window is displayed that allows you to enter either HH:MM (hours and minutes) or HH:MM:SS (hours, minutes, and seconds). If you press **<Enter>** without entering a time, the request will be canceled. Valid values for the hour are the numbers 00 through 23. Valid values for the minute are the numbers 00 through 59. Valid values for the second are the numbers 00 through 59. Each of these values require two digits to be entered.

3.8 Miscellaneous Functionality

There are four miscellaneous functionality buttons plus a query utility available. Table 19 describes these buttons.

Table 20: Miscellaneous Functionality Buttons

Button/Utility	Description
Find	The Microsimulation Output Viewer contains a Find utility by means of the Find button. You may enter the id of the object to find (link, node, or vehicle id), press <Enter>, then click on the button with the label matching the type of object you are searching for. If the object is found, it becomes highlighted, and centers the highlighted object on the screen. To unhighlight the object, you can click [Find] then click [Clear], or just click on the object that is highlighted. If you press <Enter> without entering an id or click [Clear], the Find Request will be canceled.
Print	By clicking [Print], copy of the current display is saved to the postscript file animate.ps, which is then sent to the printer specified in your environment variable PRINTER. If the environment variable is not set, the postscript file will not be sent to the printer, but you can always send the animate.ps file to a printer with the following UNIX command: lp -d printerName animate.ps
Options	Causes a window to open and display several of the viewing options, their possible states, and their current state. You may then change a viewing option from within this window. When the Display Mode Viewing Option is changed, the window closes automatically. In all other instances, you must click on the RETURN From Viewing Options Window area in order to close the window.
Quit	Used to exit the Microsimulation Output Viewer.

Query Utility. The Microsimulation Output Viewer contains a query utility. You may click on any object in the drawing area to get more detailed information regarding that particular object. If a node, link, box, parking place, or vehicle is selected, the object is highlighted and information about the object is displayed in the information window. Only one object can be selected at a time. Information about the selected object is updated as the timesteps change. Clicking on the selected object unselects and unhighlights it. If a node is selected and the intersection queue data file was specified, a window is displayed that lists the vehicle ids for all vehicles within the queues at that intersection during that particular timestep. This information is also updated as the timesteps change. To make the intersection queue window disappear, you must unselect the node or select another object. Vehicles, nodes, and parking places are highlighted by changing their color to the default highlight color (specified in the colors.graphics color file). Links and boxes are highlighted by turning on the edges if edges are not displayed, or by making the edges thicker if they are displayed.

3.9 Running as a Standalone

The Microsimulation Output Viewer can be run outside of the TAI. A configuration file needs to be created before starting the viewer. The Microsimulation Output Viewer executable is located in $TRANSIMS_HOME/sunos5.x/bin/animate.

3.9.1 Command Line Arguments

There are two command line arguments possible when starting up the Microsimulation Output Viewer. One enables you to specify the particular configuration file to use. The other enables you to display the vehicle ids next to the vehicle (called debug mode).

1) **-cf** <configuration file name>	Default: "./transims_animate.config"
2) **-d**	Debug: vehicle ids displayed next to the vehicle

3.9.2 Configuration File

The Microsimulation Output Viewer uses a configuration file in order to specify data files and viewing options. If you are accessing the Microsimulation Output Viewer via the TAI, the configuration file will automatically be created by the TAI using the values you set in the **Microsim Output** window under **Viewers**. Most options can be changed once the Microsimulation Output Viewer is running; however, data files cannot be changed. A sample animate configuration file can be found in
$TRANSIMS_HOME/data/ANIMATE/transims_animate.config.

The configuration file contains four major sections:

- Data files
- Viewing options
- Program constants
- Color specification files

The format of the configuration file is usually <keyword><value> except for the simulation start time, which has several values. Lines that begin with the pound (#) character are considered to be comments within the configuration file and are ignored by the Microsimulation Output Viewer.

3.9.2.1 Data Files

Up to six data files can be read by the Microsimulation Output Viewer:

- Network definition file (HOOPS Network Metafile)
- Box summary output file
- Link summary file
- Snapshot output file containing vehicles positions
- Snapshot output file containing traffic signal states
- Snapshot output file containing intersection queue information

The file names of these data files are specified within the configuration file. To specify these files, you must insert the following lines into the configuration file:

```
NETWORK_FILE_NAME                        <path>/network_filename.hmf
VEHICLE_DATA_FILE_NAME                   <path>/vehicle_data_filename
SIGNAL_DATA_FILE_NAME                    <path>/signal_data_filename
INTERSECTION_DATA_FILE_NAME              <path>/intersection_data_filename
SUMMARY_DENSITY_DATA_FILE_NAME           <path>/link_density_data_filename
SUMMARY_BOX_DENSITY_DATA_FILE_NAME       <path>/space_summary_data_filename
```

When specifying the path name of each of the files, do not include the tilde (~) character in the path. The Microsimulation Output Viewer is unable to process this.

The HOOPS Network Metafile is the only file that *must* be specified within the configuration file. The other files need not be specified in order to run the Microsimulation Output Viewer (although animation may be rather boring). The Network Metafile is currently generated by the ArcView Input Editor in developer's mode **Network->Export To HOOPS** command. The Network Metafiles for the TRANSIMS Release 1.0 are in the $TRANSIMS_HOME/data/ANIMATE/Networks directory.

The Vehicle, Signal, Intersection, and Box Summary data files are all generated from a Microsimulation run's output by the Simulation Output Subsystem program DumpStorage and then sorted depending on file type. The Microsimulation Output Viewer is not able to view the outputted time summary file at this time.

The Link Summary file must be user generated. There is currently no capability to create the file, so it is up to the user to determine how to produce this file. The format for the link summary file is

TIME (float) NODE (int) LINK (unsigned long) DENSITY (float).

For more information on the other file formats, see the Simulation Output Subsystem documentation.

3.9.2.2 Viewing Options

There are eight viewing options that can be set in the configuration file:

Display Mode – the DISPLAY_MODE keyword allows you to specify which data files to display. The three possible options are

1. Displaying snapshot data (DISPLAY_VEHICLES)
2. Displaying box summary data (DISPLAY_BOX_DENSITY)
3. Displaying link density data (DISPLAY_LINK_DENSITY)

Vehicle Color Option – the VEHICLE_COLOR_OPTION keyword allows you to specify how to color vehicles. The four possible options are:

1) by a default color (COLOR_BY_DEFAULT)

2) by the vehicle's status value (COLOR_BY_STATUS)

3) by a random color (COLOR_BY_RANDOM)

4) by a mapping of vehicle id to vehicle color (COLOR_BY_FILE_SPECIFICATION)

Display Parking Option – the DISPLAY_PARKING_OPTION keyword allows you to specify whether to display parking places or not. The two possible values are:

1) DISPLAY_PARKING_NO

2) DISPLAY_PARKING_YES

Display Nodes Option -- the DISPLAY_NODES_OPTION keyword allows you to specify whether to display nodes or not. The two possible values are:

1) DISPLAY_NODES_NO

2) DISPLAY_NODES_YES

Link Edge Visibility Option – the LINK_EDGE_VISIBILITY_OPTION keyword allows you to specify whether to display link and box outlines. You see more detail when zoomed out further by not displaying the link edges. The two possible values are:

1) LINK_EDGES_VISIBLE

2) LINK_EDGES_NOT_VISIBLE

Vehicle Edge Visibility Option – the VEH_EDGE_VISIBILITY_OPTION keyword allows you to specify whether to display vehicle edges. If viewing vehicles, you see the vehicle color better if not displaying vehicle edges. The two possible values are:

1) VEH_EDGES_VISIBLE

2) VEH_EDGES_NOT_VISIBLE

Signal Option – the SIGNAL_OPTION keyword allows you to specify whether the signals should all be drawn as circles (CIRCLES_ONLY) or to be drawn with arrows for protected states (ARROWS_FOR_PROTECTED).

Comment – the COMMENT keyword is used to specify a string that you wish to be displayed in the comment window of the Microsimulation Output Viewer.

3.9.2.3 Program Constants

Four program constants can be specified in the configuration file:

- EVOL_REPORT_INTERVAL
- SUMMARY_SAMPLE_INTERVAL
- SUMMARY_REPORT_INTERVAL
- SIM_START_TIME

Evolution/Snapshot Report Interval – the EVOL_REPORT_INTERVAL keyword specifies the time between output of snapshot data (in seconds) for the particular data being viewed. It is specified as an integer and defaults to 1 second. This value is used to update the clock when displaying snapshot data. This value is used when the **Display Mode Viewing Option** is set to DISPLAY_VEHICLE.

Summary Sampling Interval – the SUMMARY_SAMPLE_INTERVAL keyword specifies the time between the collection of data within the microsimulation (in seconds) for the particular data being viewed. It is specified as an integer and defaults to 1 second. This value is used when the DISPLAY_MODE Viewing Option is set to DISPLAY_BOX_DENSITY.

Summary Report/Collection Interval – the SUMMARY_REPORT_INTERVAL option specifies the time between output of summary data (in seconds) for the particular data being viewed. It is specified as an integer and defaults to 1 second. This value is used when displaying summary data.

Simulation Start Time – the SIM_START_TIME option specifies the time the microsimulation starts for the particular data being viewed. Time 0 corresponds to time SIM_START_TIME. This is used to display the animation time. The format for the time values in the configuration file are: *month day year hour minute second*. The following is an example line in the configuration file for a SIM_START_TIME of 5:00 am December 25th 1996:

```
SIM_START_TIME 12 25 1996 5 0 0
```

3.9.2.4 Color Specification Files

Four possible color specification files can be set in the configuration file.

1) Graphics color file – specifies the default colors of the objects within the Microsimulation Output Viewer, such as the links, nodes, vehicles, etc.
2) Vehicle Status color file – specifies the mapping between vehicle status values and vehicle color when the Viewing Option VEHICLE_COLOR_OPTION is set to COLOR_BY_STATUS and the DISPLAY_MODE is set to DISPLAY_VEHICLES.
3) Vehicle Specification color file – specifies the mapping between vehicle id and vehicle color when the Viewing Option VEHICLE_COLOR_OPTION is set to COLOR_BY_FILE_SPECIFICATION and the DISPLAY_MODE is set to DISPLAY_VEHICLES.
4) Link Density color file – specifies the mapping between density ranges and link or box color when displaying summary data.

The file names of these data files are specified within the configuration file. To specify these files, you must insert the following lines into the configuration file:

```
HOOPS_COLOR_FILE_NAME                    <path>/graphics_color_filename

VEH_STATUS_COLOR_FILE_NAME               <path>/veh_status_color_filename

VEH_COLOR_SPECIFICATION_FILE_NAME        <path>/veh_spec_color_filename

LINK_DENSITY_COLOR_FILE_NAME             <path>/link_density_color_filename
```

When specifying the path name of each of the files, do not include the tilde (~) character in the path. The Microsimulation Output Viewer is unable to process this.

All four color files have several characteristics in common. In all color files, the pound (#) character found at the beginning of a line marks the line as a comment and is ignored by the Microsimulation Output Viewer. Colors in all four files can be specified by either a HOOPS color string or by a hex RGB value of the format #RRGGBB. HOOPS pre-defines 72 basic color names. To use color names that have embedded blanks, you must specify the name with a dash (-) or underscore (_) character replacing the spaces. The Microsimulation Output Viewer will translate the color string.

Table 21: Predefined HOOPS Colors

apricot	aquamarine	bittersweet	black
blue	blue green	blue grey	blue violet
brick red	brown, moose brown	burnt orange	burnt sienna
cadet blue	cerulean	copper	cornflower
cyan	dandelion	forest green	fuchsia
gold	goldenrod	green	green blue
green yellow	grey, gray	indian red	jungle green
lavender	lemon yellow	magenta	mahogany
maize	maroon	melon	midnight blue
mulberry	navy blue	olive green	orange
orange red	orange yellow	orchid	peach
periwinkle	pine green	pink	plum
purple, violet	raw sienna	raw umber	red
red orange	red violet	royal purple	salmon
sea green	sepia	silver	sky blue
spring green	strawberry	tan	tangerine
teal	thistle	turquoise blue	violet blue
white	yellow	yellow green	yellow orange

Color Graphics File – See Section 3.4.5 (Color Options) for information regarding the format of the Graphics color file and the Link Density (Density Range) color file. See Section 3.4.3 (Viewing Options) under Color Values By for more information regarding the Vehicle Status Color file and the Vehicle Specification (ID) color file.

3.9.3 Specifying the Configuration File

The configuration file can be specified in four possible ways in the order listed below.

1) The filename can be specified as a command line argument (see Section 3.9.1).

2) The environmental variable TRANSIMS_ANIMATE_CONFIG is checked for the file name. To use this method, set the environment variable by putting a line in your .cshrc file that looks like:
```
setenv   TRANSIMS_ANIMATE_CONFIG   <path>/config_filename
```

3) If a command line argument is not used and the variable TRANSIMS_ANIMATE_CONFIG is not set, the current directory is searched for the file transims_animate.config.

4) If the configuration file is still not specified, the default filenames shown in Table 22 are used for the files:

Table 22: Configuration File Default File Names

Configuration File	Default
HOOPS Network Metafile	./network.hmf
Vehicle data file	./vehicle.txt
Signal data file	./signal.txt
Intersection Queue data file	./intersectin.txt
Link Summary data file	./linksum.txt
Space Summary data file	./boxsum.txt
Graphics color file	./colors.graphics
Vehicle Status color file	./colors.vehstatus
Vehicle Specification color file	./colors.vehspec
Link Density color file	./colors.density

A sample configuration file is in $TRANSIMS_HOME/data/ANIMATE/transims_animate.config along with sample color files.

4. CALIBRATION

4.1 Introduction

The TRANSIMS calibration suite is used to understand fundamental traffic flow characteristics of the TRANSIMS microsimulation. In order to determine the effects of driving rules, the calibration suite provides controlled tests of traffic flow behavior. The test networks are simplified situations where elements of the microsimulation can be tested in isolation.

4.2 Calibration Networks

The calibration networks provide the following test cases for evaluation of driving logic:

- One-lane freeway traffic to test if car-following behavior generates reasonable fundamental diagrams.
- Three-lane freeway traffic to test if passing-lane-changing behavior generates reasonable fundamental diagrams. Lane usage can also be evaluated.
- Merging at stop and yield signs and left turns against opposing traffic to test for acceptable flow rates at non-signalized intersections.
- Signalized intersection to evaluate flow rates and to test lane-changing behavior for plan following.

The freeway network (one and three lanes) is a 1000-cell (75 km) circle where vehicles enter on one side of the circle (cell 1), and flow and density measurements are taken on the opposite side of the circle (cells 491 - 495). The vehicle density is continuously increased from 0 to 0.5 (0 veh/km/lane to 66 veh/km/lane) during the simulation run.

To test merge behavior at stop and yield signs at an unsignalized intersection, an incoming link with one lane is added to the circle at cell 500. The incoming vehicles are removed at site 750.

To test left turns against opposing traffic, two links are added to the circle. An opposing link that ends at cell 500 on the circle provides vehicles that will make a left turn across the traffic on the circle. The vehicles turn across the circle onto a link that begins at cell 500 where they eventually exit from the simulation.

The signalized intersection includes a three-lane approach link. Three one-lane links exit from the intersection: one left, one straight, and one right. Incoming vehicles attempt lane changes on the approach link in order to follow their intended movement at the intersection. The intersection has a signal with a 60-second green phase and a 60-second red phase.

A detailed examination of microsimulation traffic flow characteristics using the calibration suite is in Section 4.5, *TRANSIMS Traffic Flow Characteristics*.

The driving rules used by the TRANSIMS microsimulation can be changed using the Driver Logic dialog in the TAI. The calibration suite should always be used to examine the effects of changes.

4.3 Calibration Filters

The microsimulation output files from calibration runs are filtered to simplify data analysis. The filters on the freeway, merge, and left-turn calibration runs collect summary data on five-cell sample blocks placed at specified locations on the calibration networks (Table 23). The data is summarized over three-minute intervals. The sample blocks cut across all lanes of the link, but data is reported on a lane-by-lane basis, as well as the link totals.

Table 23: Calibration Network – Sampling Locations

Network	Sample Block Location	Sampled Type
Freeway	Sites 491 - 495 on the circle	Circle vehicles
Merge	Sites 491 - 495 (Link 7) on the circle Sites 501 - 505 (Link 1) on the circle	Circle vehicles Merging vehicles
Left Turn	Sites 491 - 495 (Link 7) on the circle Sites 1 - 5 (Link 11) on exiting link	Circle vehicles Left-turn vehicles

The filter for the signalized intersection divides the 1000-meter length (133 cells) of the approach link into seven boxes of 20 cells (150 meters) and produces 30-second summary data for the boxes. Two filtered output files are produced: one containing vehicle lane-state data, and one containing vehicle turn data.

4.4 Filtered File Formats

The filtered files are tab-delimited ASCII text with one data record per line of the file. The first line of the filtered file lists the names of the data fields reported in the order that the data for those fields appears in subsequent lines (Tables 29-33). The flow and density data for freeway, merge, and left-turn calibration runs is calculated for each lane, as well as the total for all lanes.

Table 24: Freeway Filter Format

Field	Interpretation
Simulation Time	Seconds since simulation start
Lane	Lane number
Density	Vehicles/km/lane at the sample block on the circle
Flow	Vehicles/hour/lane at the sample block on the circle

Table 25: Merge Filter Format

Field	Interpretation
Simulation Time	Seconds since simulation start
Lane	Lane number
Flow-7	Flow of circle traffic in vehicles/hour/lane (Link 7)
Density-7	Density of circle traffic in vehicles/km/lane (Link 7)
Flow-11	Flow of left-turn vehicles in vehicles/hour/lane (Link 11)

Table 26: Left-turn Filter Format

Field	Interpretation
Simulation Time	Seconds since simulation start
Lane	Lane number
Flow-7	Flow of circle traffic in vehicles/hour/lane (Link 7)
Density-7	Density of circle traffic in vehicles/km/lane (Link 7)
Flow-11	Flow of left-turn vehicles in vehicles/hour/lane (Link 11)

Table 27: Signalized Intersection Filter -- Vehicle Lane State

Field	Interpretation
Time	Seconds since simulation start
Box Distance From Node 2	Starting distance of box in meters measured from the node from which the vehicles are traveling away
Light Color	State of traffic control (g=green, r=red)
Light Cycle	Length of traffic control cycle in seconds
#in-1-to-1	Number of vehicles in the sample box during the 30-second interval that were in lane 1 and planned to turn left at the intersection
#in-1-to-2	Number of vehicles in the sample box during the 30-second interval that were in lane 1 and planned to go straight at the intersection
#in-1-to-3	Number of vehicles in the sample box during the 30-second interval that were in lane 1 and planned to turn right at the intersection
#in-2-to-1	Number of vehicles in the sample box during the 30-second interval that were in lane 2 and planned to turn left at the intersection
#in-2-to-2	Number of vehicles in the sample box during the 30-second interval that were in lane 2 and planned to go straight at the intersection
#in-2-to-3	Number of vehicles in the sample box during the 30-second interval that were in lane 2 and planned to turn right at the intersection
#in-3-to-1	Number of vehicles in the sample box during the 30-second interval that were in lane 3 and planned to turn left at the intersection
#in-3-to-2	Number of vehicles in the sample box during the 30-second interval that were in lane 3 and planned to go straight at the intersection
#in-3-to-3	Number of vehicles in the sample box during the 30-second interval that were in lane 3 and planned to turn right at the intersection

Table 28: Signalized Intersection Filter -- Vehicle Turn Data

Field	Interpretation
Time	Seconds since simulation start
Light Cycle	Length of traffic control cycle in seconds
#left turns	Number of vehicles that turned left during the 30-second interval
#ahead	Number of vehicles that went straight during the 30-second interval
#right turns	Number of vehicles that turned right during the 30-second interval
#total lost	Number of vehicles that were off-plan during the 30-second interval
#lost with plan to turn left	Number of vehicles that were off-plan and planned to turn left during the 30-second interval
#lost with plan to go straight	Number of vehicles that were off-plan and planned to go straight during the 30-second interval
#lost with plan to turn right	Number of vehicles that were off-plan and planned to turn right during the 30-second interval

4.5 TRANSIMS Traffic Flow Characteristics

[Input for this section if from the preprint of a paper entitled "TRANSIMS traffic flow characteristics", by K. Nagel, P. Stretz, M. Pieck, S. Leckey, R. Donnelly, and C. L. Barrett. The paper was presented to the Transportation Research Board in Washington, D.C., in January 1998.]

4.5.1 Introduction

One could probably reach agreement that the traffic flow behavior of traffic simulation models should be well documented. Yet, in practice, this turns out to be somewhat difficult. Many traffic simulation models are under continuous development, and the traffic flow dynamics documented in a certain publication is often a *snapshot*, valid at the time of writing, but no longer the true state of the model.

It thus makes sense to agree on a certain set of tests for traffic flow dynamics which should be run and documented together with *real* results. In this paper, we propose a (probably incomplete) suite of traffic flow measurements. Also, some of the results in this paper are arguably unrefined with respect to reality. Yet, as we stated above, we are continuously working on improvements, and this publication represents both a snapshot of where we currently stand and an argument for a standardized traffic flow test suite for simulation models. We hope that this publication will both open the way for a constructive dialogue on which standardized traffic flow tests should be run for traffic simulation models, and which of the features of our traffic simulation models may need improvement.

This paper starts with a general section on validation and calibration (Section 4.5.2), followed by a high-level description of the TRANSIMS microsimulation approach (Section 4.5.3). Section 4.5.4 is a fairly technical description of the actual implementation. Section 4.5.5 contains a description of the test cases that we ran for this paper and presents the simulation results. Section 4.5.6 contains an example of parameter sensitivity testing for the case of a yield sign, followed by a short section outlining differences in the logic when the simulation was used for the so-called Dallas case study (Section 4.5.7). The paper is concluded by a discussion section (Section 4.5.8) and a summary (Section 4.5.9).

4.5.2 Validation, Calibration, etc.

Prerequisite of any simulation model to be used is a certain amount of confidence in its output. The process of building confidence depends on human nature and is sometimes hard to explain. Yet, an organized process toward model acceptance would help. Such an acceptance process may be composed of the following four elements [1].

1) Verification – have the hypothesized behavioral rules been implemented correctly?

2) Validation – do the hypothesized behavioral rules produce correct emergent behavior, such as correct fundamental diagrams? Note that this does not specify a quantitative procedure; plausibility, consistency with theory and experience, and documentation of emergent behavior are the important elements here.

3) Calibration – have the model parameters been optimized to (possibly site specific) settings? This requires a decision on a data set and a decision on an objective function that can *quantify* the closeness of the simulation to the data set.

4) Accreditation – given a question, is the model indeed powerful enough to help with it?

Note that this process is not uni-directional. For example, if one cannot calibrate a model very well for a given scenario and a given objective function, one will go back and change the microscopic rules and then have to go through verification and validation again.

Also, a formally correct verification process can be shown to be mathematically hard or computationally impossible except in very simple situations (see, e.g., Chapters 14 - 16 in [2]). Intuitively, the problem is that seemingly unrelated parts of the implementation can interact in complicated ways, and to exhaustively test all combinations is impossible. For that reason, both practitioners and theoreticians suggest that one needs to allocate resources intelligently between verification and validation.

Sometimes, the word *validation* is also used when a simulation model, *after* calibration to a scenario and data set A, is run under another scenario to test its predictive performance. Since this represents, in principle, the same procedure – run the simulation model against a scenario without further adjustment in the process – we do not see a problem in the use of the word validation in both cases.

Next, one needs to decide on which networks to run the above studies. The following seem to be useful:

- **Building block cases** such as *traffic in a loop* or *traffic through yield sign*. The chapters of the Highway Capacity Manual [3], despite being under discussion, seem to be a good starting point here. Maybe these cases will not be very useful for calibration since *clean* data on these cases is difficult, if not impossible, to obtain. Yet, these cases would certainly allow plausibility check of a simulation model and comparison to other simulation models.

- **Complicated test cases**, which test a variety of behavior such as merging or traffic signals, in a larger *context* (i.e., when interacting). It would be nicest to have test cases from the real world, together with real data. These test cases would best be made electronically available.

Of course, models have always been validated and calibrated (e.g., [4-6]). For fluid-dynamical models, calibration can be formalized [7,8]. Yet, we would like to stress that there are two diverging tendencies here:

- Models that are simple (i.e., have few parameters) are easy to be formally calibrated in the sense that one can adjust the parameters so that some objective function is minimized. Yet, the model may be too simple to indeed reflect the *meaning* of the data.[1]

- Models that have many parameters are in principle capable of representing a much wider variety of dynamics. Yet, they are difficult for formal calibration because the degrees of freedom are too large. Here, the intuition of the developer is important, who prescribes the simplifications, usually by making the problem more homogeneous than it is (for example

[1] Bluntly, one can always fit a straight line to a data cloud.

prescribing that drivers only fall into few behavioral classes). Microscopic models fall into this category.

Ref. [9] nicely illustrates the problem: The authors indeed decide on an objective function (match the two parameters of a two-fluid model description of the real world traffic); yet the procedure is trial and error in the sense that the authors themselves decide on which aspects of NETSIM they believe to be important.

This indicates, consistent with our own experience, that formal calibration (in the sense of a formal procedure as opposed to trial-and-error) of *microscopic* models is currently very hard to achieve. This, in addition to the generally valid argument that calibration does not protect one against having the wrong model, implies to us that on the *validation* level, comparable and meaningful test suites should be constructed, and that the model behavior in these test suites should be publicly documented. This effort should be geared towards *understanding* the strength and weaknesses of a/the participating model (as opposed to deciding which is the *best* model).

In this paper, we want to concentrate on the *validation* part in the above sense in conjunction with *building block* test cases. We mean that as a first important step; in the future, we would like to be able to say something like "the simulations in this study are based on driving rules with their emergent behavior documented in the appendix", which would recognize the fact that the rules may have changed since the last *major* publication. This does not preclude that we will attempt to construct more realistic test scenarios in the future.

4.5.3 The TRANSIMS Microsimulation Approach

When designing a traffic microsimulation model, the first idea might be to measure all aspects of human driving and put them in algorithmic form into the computer. Unfortunately, such attempts cause many problems. The first is a data collection problem, because one can certainly not measure *all* aspects of human driving and is thus faced with the double-sided problem that the necessary data collection process is extremely costly and still selective. Second, what if the emergent flow properties of such a model are clearly wrong; for example, producing an hourly flow rate that is much too high?

For that reason, the TRANSIMS [10] microsimulation starts with a *minimal* approach. A minimal set of driving rules is used to simulate traffic, and this set of rules is only extended when it becomes clear that a certain important aspect of traffic flow behavior cannot be modeled with the current rule set.[2] Besides the conceptual clarity, this also has the advantage that it is usually computationally fast – minimal models have few rules, and thus run fast on computers.

The last paragraph leaves open what the *important aspects* are. In our view, this can only be decided in the proper context, i.e., when the question or problem area of application is known. The questions that TRANSIMS is currently designed for are transportation *planning* questions. These questions have traditionally been approached using traffic assignment models based on link performance functions (link capacity functions). Link performance functions are known to be dynamically wrong in the congested regime [11]; they simply do not model queue build-up when demand is higher than capacity.

[2] Note, though, that it is certainly desirable to have *reasonable* microscopic rules.

The most important result of a transportation microsimulation in that context should be the *delays*, since they dominate travel times, and also hinder discharge of the transportation system, thus leading to grid-lock. Delays are caused by congestion, and congestion is caused by demand being higher than capacity. This implies that the first thing the TRANSIMS traffic microsimulation has to get right are capacity constraints (and possibly their variance). Capacity constraints are caused by a variety of effects:

- Undisturbed roadways such as freeways have capacity constraints given by the maximum of the flow-density diagram.
- Typical arterials have their capacity constraints given by traffic lights.
- In the case of unprotected turning movements (yield, stop, ramps, unprotected left, etc.), the capacity constraints are given as a function of opposing traffic flows. For example, the number of vehicles making an unprotected left turn depends on the oncoming traffic.

Building a simulation that can be adjusted against all of these diagrams seems to be a hopeless task given the enormous amount of degrees of freedom. The TRANSIMS approach, for that reason, has been to *generate* the correct behavior from a few much more basic parameters. The correct behavior with respect to the above criteria can essentially be obtained by adjusting two parameters:

1) The value of a certain asymmetric noise parameter in the acceleration determines maximum flow on freeways and through traffic lights.
2) The value of the gap acceptance determines flow for unprotected movements.

It must be emphasized again that these remarks are only valid in our context: There are many questions for which the models need to have a higher fidelity, and then more details, higher resolution, etc., may need to be added (e.g., [12, 13]).

There is sometimes debate whether the model we thus obtain is truly *microscopic*. We use the term *microscopic* with respect to the *resolution* of the model; i.e., a model is microscopic as soon as it allows the identification of individual particles (here cars). The proposed area of *application*, though, is where traditionally more macroscopic models have been used [11, 14-16].

4.5.4 Rules of the Model

4.5.4.1 Single Lane Uni-directional Traffic

Our traffic simulation is based on a cellular automata technique, i.e., a road is composed of cells, and each cell can either be empty, or occupied by exactly one vehicle [17, 18], see Figure 31 (a). Since movement has to be from one cell to another cell, velocities have to be integer numbers between 0 and V_{max}, where the unit of velocity is [cells per timestep]. It turns out that reasonable values are [18, 19]:

- length of a box = $1/r_{jam}$ = 7.5m (r_{jam} = density of vehicles in a jam)
- timestep = 1 sec
- maximum velocity = 5 boxes per timestep = $5 \cdot 7.5$ m/sec = 135 km/h ~ 85 mph

For other conditions, such as higher or lower speed limits, this can be adapted.

Note that this approach implies a *coarse graining* of the spatial and temporal resolution and, therefore, of the velocities. A vehicle that has a speed of, say, 4 in this model stands for a vehicle that has a speed anywhere between $3.5 \cdot 7.5$ meters/sec ~ 95 km/h (59 mph) and $4.49999 \cdot 7.5$ meters/sec ~ 121 km/h (75 mph).

Vehicles move only in one direction. For an arbitrary configuration (velocity and position), one update of the traffic system consists of two steps: a velocity update step consisting of three consecutive rules, and a movement step according to the result of the velocity update. The whole update is performed simultaneously for all vehicles. The complete configuration at timestep *t* is stored, and the configuration at timestep *t+1* is computed from that *old* information. Computationally, we calculate in timestep *t* (with the three rules) the new velocity of each car and write this newly calculated velocity in the same site without moving the car (velocity update). After that, we move all cars according to their newly calculated velocity (movement update).

1) (velocity update)

For all particles *i* simultaneously, do the following:

```
IF $(v_i \geq gap_i)$

      $v_i := \begin{cases} gap_i - 1 \text{ with probability } p_{noise} \text{ if possible}^{3} \\ gap_i \quad \text{else} \end{cases}$      (close following/braking)

ELSE IF $(v_i < v_{\max})$

      $v_i := \begin{cases} v_i \quad \text{with probability } p_{noise} \\ v_i + 1 \quad \text{else} \end{cases}$      (acceleration)

ELSE (i.e. $(v_i = v_{\max}$ AND $v_i < gap_i))$

      $v_i := \begin{cases} v_{\max} - 1 \text{ with probability } p_{noise} \\ v_{\max} \quad \text{else} \end{cases}$      (free driving)

ENDIF
```

2) (movement update)
   Move all particles *i* to $x_i(t+1) = x_i(t) + v_i$.

The index *i* denotes the position (an integer number) of a vehicle, *v(i)* its current velocity, v_{max} its maximum speed, *gap(i)* the number of empty cells ahead, and p_{noise} is a randomization parameter.

The first velocity rule represents noisy car following or braking. If the vehicle ahead is too close, the vehicle itself attempts to adjusts its velocity such that it would, in the next timestep, reach a position just behind where the vehicle ahead is at the moment. Yet, with probability p_{noise}, the vehicle is a bit slower than this.

The second velocity rule represents noisy acceleration. Essentially, the acceleration is linear (i.e., independent from current speed), but with probability p_{noise}, no acceleration happens in the current timestep (maybe as a result of switching gears etc.). Instead of an acceleration sequence of $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \ldots$, a possible acceleration sequence can now be $0 \rightarrow 0 \rightarrow 1 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 3 \ldots$.

The last velocity rule represents free driving. Instead of remaining always at the same speed, such vehicles fluctuate between v_{max} (with probability $1 - p_{noise}$) and $v_{max} - 1$ (with probability p_{noise}). Note that a vehicle that is set to $v_{max} - 1$ will go through the acceleration step next time; thus in the next timestep, either staying at $v_{max} - 1$ with probability p_{noise} or getting back to v_{max}. Note that the resulting average speed of a freely driving vehicle is thus $v_{max} - p_{noise}$.

In terms of a microscopic foundation, the model is composed of the following elements:

- If a vehicle does not have enough space ahead, speed is proportional to space headway, which implies constant time headway (Pipes' theory, [20]).
- If there is enough space ahead for the given velocity, the vehicle accelerates linearly either to maximum speed or until the space headway becomes too small for further acceleration. A more realistic acceleration would probably be proportional to $1/v$, where v is the speed. This would be computationally more burdensome; nevertheless, studies about the effect are under way. The effect on the principal traffic dynamics seems to be minimal [21].
- Deceleration is instantaneous within one timestep. If one wants to constrain the model to realistic deceleration values, one needs to look at the velocity of the vehicle ahead. This is again computationally more burdensome, and the precise difference when changing this element alone with respect to the traffic dynamics is unclear [22], although it is clear that throughput will go up [23].
- On top of these rules, we add a fairly large amount of random noise in the velocity decision.

Somewhat shorter, the model enforces constant time headway for close following and for braking, but acceleration is *delayed*. This puts the model into a large class of dynamically similar models that use *time delayed* constant time headway; e.g., of the type $a(t) \alpha V [\Delta x(t)] - v(t)$ or $v(t + t) \alpha V [D x(t)]$, where a is the acceleration, v is the vehicle's velocity, Dx is the space headway, $V(Dx)$ is a desired speed function, and t is a time delay. It is certainly arguable that this does not catch all aspects of traffic; yet, all of these models are remarkably robust with respect to their traffic dynamics behavior, both in microscopic [22, 24, 25] and in fluid-dynamical [26, 27] implementations.

4.5.4.2 Lane Changing for Passing

For multi-lane traffic, the model consists of parallel single lane models with additional rules for lane changing. Here we describe the two-lane model, which can be modified to any kind of multi-lane model. Lane changing is modeled by an additional update step, which is added before the velocity update. The new sequence of steps is presented below. Steps two and three are the same in the single-lane model, and they are executed separately for each lane.

1) Lane-changing decision

2) Velocity update

3) Vehicle movement

According to this lane-changing rule set, the vehicles are only moving sideways during the lane-changing step; forward movement is done in the vehicle movement step. One should, though, look at the combined effect of the lane changing and vehicle movement, and then vehicles will usually have moved sideways *and* forward. The decision to change lane is implemented as strictly parallel update; i.e., each vehicle is making its decision based on the configuration at the beginning of the update.

```
• Lane-changing decision for passing
    – IF neighboring position $x_o(i)$ in other lane is vacant
        * THEN Calculate:
            • $gap(i)$ Gap Forward in Current Lane,
            • $gap_o(i)$ Gap Forward in Other Lane,
            • $gap_b(i)$ Gap Backward in Other Lane,
            • IF ( $gap(i) < v(i)$ AND $gap_o(i) > gap(i)$ )
                – THEN $weight1 = 1$
                – ELSE $weight1 = 0$
            • $weight2 = v(i) - gap_o(i)$
            • $weight3 = v_{max} - gap_b(i)$.
        * IF ( $weight1 > weight2$ ) AND ( $weight1 > weight3$ )[3]
            • THEN mark vehicle for lane change[4]
```

The rules are working in the following way (see Figure 31 (b)): First we look at the neighboring position in the target lane. If this cell is vacant, we calculate the gap forward in the current lane (*gap*), the gap forward in the target lane (gap_o), and the gap backward in the target lane (gap_b). With these results, we calculate the *weight1* to *weight3* described above. Finally, if the weight comparisons render true, the car will change to the new lane. After executing the lane-changing decision, we calculate the new velocity for all cars and move them according to this velocity.

This lane-changing implementation follows a usual structure [29, 30].

- Reason to change lanes? (Slow car ahead? Need to make turn later (see below)?)
- If yes: Target lane empty? (Definition of *empty* depends on *urgency*)

[3] Weights are used because of extensibility toward "lane changing for plan following". See below.
[4] In the current version, the lane change is actually still rejected with a probability of 0.01 even when all rules are fulfilled. This is in order to break the following artifact or variations of it: Assume one lane is completely occupied and one is completely empty. The above rule set will result in these vehicles just changing back and forth between the lanes—the vehicles will never get smeared out across the lanes. See Ref. [28] for more details.

- If yes: change lanes except for stochastic noise

Lane change implementations using this framework are remarkably robust in their dynamic behavior [29-32]. This allows us, for example, not to look at other vehicle's velocities: The forward condition for the target lane, gap_o ³ v, is consistent with the condition v £ gap for the car following; the backward condition for the target lane, gap_b ³ v_{max} is simply a worst case scenario which, nevertheless, does not perform, in the analysis of the emergent properties, any worse than a condition that depends on the velocity of the other car (compare, e.g., [33] with [34]).

For three or more lanes, a simultaneous implementation of the lane-changing decision can lead to collisions. For example, in a three-lane road, two vehicles on the left and right lane could decide to go to the same spot in the middle lane. From an algorithmic point of view, this is possible because the lane-changing decision is based on the configuration on time *t*; but it is also an entirely realistic situation.[5] To avoid collision, we only allow lane changes in a certain direction in each timestep:

- IF the timestep is even

  THEN start procedure *lane-changing decision* to the *left* for cars on the middle and then on the right lane

- IF the timestep is odd

  THEN start procedure *lane-changing decision* to the *right* side for cars on the middle and then on the left lane

Thus, left lane changes occur only on even timesteps; right lane changes occur only on odd timesteps. This behavior is collision free.

4.5.4.3 Lane Changing for Plan Following

Vehicles in TRANSIMS follow route plans; i.e., they know ahead of time the sequence of links they intend to follow. This means that, when they approach an intersection, they need to move into the correct lanes in order to make the intended turn. For example, a vehicle that intends, according to its route plan, to make a left turn at the next intersection must move into one of the lanes that actually allow a left turn.

This is achieved in TRANSIMS by supplementing the basic lane-changing rules with a bias toward the intended lanes. This bias increases with increasing urgency; i.e., with decreasing distance to the intersection. Technically, this is achieved by adding another weight to the acceptance conditions for lane changing:

- IF (*weight1* + *weight4* > *weight2*) AND (*weight1* + *weight4* > *weight3*)

  THEN change lane

weight4 is calculated according to

[5] In a deeper sense, the problem is caused by the fact that the underlying decision-making dynamics has a time scale that is smaller than the time resolution of the simulation. The simulation thus must resolve the conflict by other means [35].

$$weight4 = \max\left[\frac{d^* - d}{v_{\max}}, 0\right]$$

for lane changes in the desired direction as long as the vehicle is not in one of the correct lanes, cf. Figure 31 (c). *d* is the remaining distance to the intersection, *d** is a parameter; both are given in the unit of *cells*. *d** is currently set to 70 cells; i.e., approx. 500 m or 1/3 of a mile, throughout the simulation. In consequence, *weight4* increases from zero to $d^* / v_{max} = 14$ during the approach to the intersection. If $weight4 = 0$, then it does not influence lane changing decision. $weight4 = 1$ has the same effect as a slower vehicle ahead on the same lane. Further increases of *weight4* more and more override the security criteria that the forward and the backward gap on the destination lane need to be large enough. $weight4 > v_{max}$ lets the vehicle make the lane change even if only the neighboring cell on the destination lane is free.

Once a vehicle is in one of the *correct* lanes within 70 cells (525m) of the intersection, it is only allowed to change lanes if the target lane is also *correct*. For movements that are allowed on multiple lanes through the intersection, this leads to equal usage of these lanes. This algorithm is *not* capable of leaving a single *correct* lane temporarily when encountering, say, a stopped bus on the same lane.

4.5.4.4 Unprotected Turning Movements

A necessary element of traffic simulations is unprotected turning movements. By this we mean that that for the movement the driver intends to make, some other lanes have priority. Examples are stop signs, yield signs, on-ramps, and unprotected left turns.

The general modeling principle for this in TRANSIMS is based on a gap acceptance in the opposing (in TRANSIMS sometimes called *interfering*) lanes, see Figure 31 (d). Opposing lanes are the lanes that have priority; for example, for a stop-controlled left turn onto a major road, this would be all lanes coming from the left plus the leftmost lane coming from the right. In order to accept the turn, there has to be a sufficient gap in each of these lanes.

Note that "gap divided by the velocity of the oncoming vehicle" is the oncoming vehicle's time headway, so the dynamics of this follows the Highway Capacity Manual [3]. If one wants a time headway on an opposing lane of at least three seconds, then a vehicle with a velocity of 4 cells/second would have to be at least 12 cells away from the intersection.

The current TRANSIMS microsimulation uses a gap acceptance (gap between intersection and nearest car to the intersection that is approaching) of three(3) times the oncoming vehicle's velocity; i.e., when the gap on each opposing lane is larger than or equal to the first vehicle on that lane, the move is accepted. For example, if the oncoming vehicle has a speed of 3, at least nine (9) empty cells have to be between the oncoming vehicle and the intersection. A special case is if the oncoming vehicle has the velocity zero, in which case no gap is necessary.

4.5.4.5 Signalized Intersections

In TRANSIMS, we distinguish between signalized intersections and unsignalized intersections. In signalized intersections, the priorities are changing in time and regulated by signals. In unsignalized intersections, the priorities are fixed.

When a simulated vehicle approaches a *signalized* intersection, the algorithm first decides if, according to its current speed, it potentially wants to leave the link; i.e., its current speed (in cells per update) is larger than or equal to the remaining number of cells on the link.[6] If a vehicle wants to leave the link, the algorithm checks the *traffic control,* which determines if the vehicle can leave the link. If it encounters a red light, it cannot leave the link and no further action is taken. If it encounters a protected (green arrow) or caution (yellow) signal, the vehicle is allowed to enter the intersection. If it encounters a permitted signal (green, for example permitted left turn against oncoming traffic), the vehicle checks all opposing flows for a gap that is larger or equal to three (3) times the oncoming vehicle's velocity (see Section 4.5.4.4).

If the movement into the intersection is accepted, the vehicle is moved into an *intersection queue*; there is one queue for each incoming lane. This queue models vehicle behavior inside an intersection. The vehicle gets a *time stamp*, before which it is not allowed to leave the intersection; this time stamp is representative of the duration of the movement through the intersection. The intersection queues have finite capacity; once they are full, no more vehicles are accepted and the vehicles start to queue up on the link. This models the finite vehicle storing capacity of an intersection.

Once a vehicle is ready to leave the intersection, it moves to the first cell on the destination link, if available. The speed of the vehicle is not changed when it is in the intersection queue so it exits on the destination link in the first cell with the same velocity that it had when it entered the queue.

Note that vehicles turning against opposing traffic make their decision to accept the turn when they *enter* the intersection queue, not when they leave it. This can have the effect that a vehicle enters the intersection queue when there is no oncoming traffic; but, because of other vehicles ahead of it in the same queue, cannot make its turn immediately. Yet, since the turn was already accepted, it will be executed as soon as all vehicles ahead in the same queue have cleared the queue and a cell on the destination link is available. The turn can occur during oncoming traffic. So in some sense, vehicles will go *through* each other. Yet, note that on average, the result is still correct. The approach described above will not let more vehicles through the intersection than a gap acceptance calculated when *leaving* the intersection queue. The above logic was chosen for simplification purposes since unsignalized intersections (see below) do not have queues and thus *must* make their acceptance decisions when entering the intersection.

4.5.4.6 Unsignalized Intersections

Unsignalized intersections in TRANSIMS have no internal queues; i.e., vehicles go right through them.[7] Also, vehicles leaving an unsignalized intersection go down the destination link as far as prescribed by their velocity, not just into the first cell as in the signalized intersections. Apart from these two differences, unsignalized intersections are similar to signalized ones.

When a simulated vehicle approaches an unsignalized intersection, the algorithm first decides if, according to its current speed, it potentially wants to leave the link; i.e., its current speed (in cells per update) is larger than or equal to the remaining number of sites on the link. If a vehicle wants

[6] Vehicles may accelerate or slow down before they actually reach the intersection. See below.
[7] Again, technically the vehicles only reserve cells on the destination links. The actual move through the intersection happens later and can also be postponed if, after the velocity update, the vehicle actually does *not* make it to the intersection.

to leave the link, the algorithm checks the *traffic control*, which determines if the vehicle can leave the link. Currently, occurring traffic controls are: no control, yield, and stop.

If a *no control* is encountered, the vehicle is moved to its destination cell without any further checks. For example, if a vehicle has a velocity of five (5) cells per update and two (2) more cells to go on its link, then it attempts to go three (3) cells into the destination link. If that cell is already reserved (either by another *reservation* or by a real vehicle), then the next closer cell is attempted, etc., until the algorithm either finds an empty cell or returns that the destination lane is full. *No control* is usually used for the major directions; i.e., for the lanes that have priority.

If a *yield sign* is encountered, the vehicle checks the gap on all opposing lanes. According to the same rules as above, on all opposing lanes the gap must be larger or equal to three times the first vehicle's speed on that lane. If the movement is accepted, the destination cell is selected according to the same rules as with the *no control* case.

If it encounters a *stop sign*, the vehicle is brought to a stop. Only when the vehicle has a velocity of zero for at least one timestep on the last cell of the link is it allowed to continue. If the result of the regular velocity update indeed accelerates the vehicle,[8] then it attempts to go through the intersection. On all opposing lanes, the gap, according to the same rules as above, must be larger or equal to three times the first vehicle's speed on that lane. If the movement is accepted, a vehicle coming from a stop sign will always go to the first cell on the destination link (if empty) and will have a velocity of one.

4.5.4.7 Parking Locations

In the current TRANSIMS microsimulation, vehicular trips start and end at parking locations. Each link in the microsimulation, except for freeway ramps, freeway links, and some *virtual* links such as centroid connectors, has at least one parking location. Parking locations thus represent the aggregated parking options on that link. Parking locations have rules about how vehicles enter and exit the simulation:

- Each vehicle in TRANSIMS has a complete route plan, together with a starting time. At the starting time, the vehicle is added to a queue of vehicles that want to leave the same parking location. When the vehicle is the first one in the queue, it attempts to enter the link. The acceptance logic is, in spirit, similar to the logic of the unsignalized intersections; i.e., vehicles check the available gap and make their decision based on that. Parking accessory logic is not the focus of the current paper, and since that logic may change in TRANSIMS in the near future and we also expect no influence on the results presented here, we omit further technical details.
- A vehicle that has reached its destination parking location according to its plan will leave the microsimulation.

4.5.4.8 Parallel Logic

TRANSIMS is designed to run on parallel computers, such as coupled workstations, desktop multiprocessors, or supercomputers. The parallelization approach used for the microsimulation is

[8] That is, there is a probability of 1 - p_{noise} that the vehicle will not accelerate in the given timestep.

a geographical distribution; i.e., different geographical parts of the simulated area are computed on different CPUs.

The current TRANSIMS microsimulation has these boundaries always in the middle of links. This is done in order to keep the complexity of the parallel computing logic as far away as possible from the complexity of the intersection logic.

Information must be exchanged at the boundaries several times per update in order to keep the dynamics consistent. For example, if a vehicle changes lanes and ends up close in front of another one, that other one is probably forced to brake. Now, if the lane-changing vehicle is on one CPU and the following one on another, one needs to communicate the lane change. This will be called *Update boundaries* in the following section.

4.5.4.9 Complete Scheduling

For a complete transportation microsimulation, we need to specify when movements are accepted, and also how conflicts are resolved. For example, vehicles simultaneously attempting to change lanes into the middle lane represent such a conflict. Another conflict is two vehicles from two different links competing for the same site on the destination link.

The complete update of the current TRANSIMS microsimulation is as follows. Assume that the state at time t is the result of the last update. Let $t1$, $t2$, etc., be intermediate partial timesteps.

1) Vehicles that are ready to leave intersection queues from signalized intersections reserve cells on outgoing lanes. They only attempt to reserve the first cell on the link; their velocity is the same as it was when they entered the intersection. When the cell is occupied (either by another *reservation* or by a vehicle), then the vehicle cannot leave the intersection. Note that there can be a conflict between different queues for the same destination cell. The current solution in TRANSIMS is that queues are served on a first-come, first-served basis in some arbitrarily defined way; i.e., a queue that happens to be treated earlier in the microsimulation has a slightly higher chance of unloading its vehicles. Result: t_1 information.

2) Vehicles change Lanes. Use information from time t_1 to calculate situation at time t_2.

3) Exit from Parking. Results in t_3 information.

4) Exchange boundary information for parallel computing.

5) Non-signalized intersections reserve sites on target lanes. Note that there can be a conflict of two incoming links competing for the same destination cell. The current solution in TRANSIMS is that links are served on a first-come, first-served basis; i.e., a link that happens to be treated earlier in the microsimulation has a slightly higher chance of unloading its vehicles. Note that this conflict only happens between minor links. Major links never compete for the same outgoing link except when there is a network coding error; and for the competition between major and minor links, the major link always wins because of the opposing lanes conditions.[9] Result: t_4 information.

[9] Note that the situation is slightly different when the speed of the vehicle on the major link is zero – see below.

6) Calculate speeds and do movements. If a vehicle scheduled for an intersection does not go through the intersection as a result of the velocity update, the reservation is canceled. Vehicles that go through unsignalized intersections have p set to zero; i.e., if it turns out that the result of the velocity update indeed brings them into the intersection, they must go to the site on the destination lane that was reserved earlier. Result: $t_5 = t+1$ information.

7) Exchange boundary information and migrate vehicles for parallel computing.

4.5.5 Toward a Standardized Flow Test Suite for Simulation Models

In order to control the effect of driving rules, TRANSIMS provides controlled tests for traffic flow behavior. These tests are simplified situations where elements of the microsimulation can be tested in isolation. This test suite uses the standard microsimulation code in the same way it is used for full-scale regional simulations, and it also uses the same input and output facilities: The test network is currently defined via a table in an Oracle database, in the same format as the Dallas-Ft. Worth network is kept. Input of vehicles is, following individual vehicle's plans, via parking locations, the same way vehicles enter regional simulations.[10] Output is collected on certain parts of the network on a second-by-second basis, the same way it can be collected for regional microsimulations. The collected output is then postprocessed to obtain the aggregated results presented in this paper.

The test cases we look at in this paper are the following (see also Figure 31 (e)):

- One-lane traffic, in order to see if car following behavior generates reasonable fundamental diagrams.
- Three-lane traffic, in order to see if the addition of passing lane changing behavior still generates reasonable fundamental diagrams, and in order to look at lane usage.
- Stop sign, yield sign, and left turns against oncoming traffic, in order to see if the logic for non-signalized intersections generates acceptable flow rates.
- A signalized intersection, in order to see if we obtain reasonable flow rates, and in order to check lane-changing behavior for plan following purposes.

4.5.5.1 Measured Quantities

We look at three-minute averages of the following quantities:

- **Flow, Volume**. Flow q is defined as usual by:

$$q = \frac{N}{T} \quad [vehicles / hour]$$

[10] Route plans are simply necessary to be consistent with the way the simulation is normally used; for the test cases, we use very few types of generic route plans (like “enter the microsimulation and keep on driving in a circle indefinitely”) and replicate them with different starting times to fulfill our needs. This is not much different from departure rates.

N is the number of cars that pass a certain site at a time period *T*.

- **Density**. Density is, in principle, easily defined, $r = N / L$, where *N* is the number of vehicles on a piece of roadway of length *L*. Yet, given current sensor technology, this is not easy to achieve since one would need a sensor that counts, say once a second, cars on a predefined stretch of length *L* of the roadway. For that reason, empirical papers sometimes resort to occupancy, which is the fraction of time a given sensor has been occupied by a vehicle. Currently TRANSIMS measures density according to its original definition; i.e., once per timestep, we count the number of vehicles on a stretch of roadway of L = 5 sites =5 x 7.5m = 37.5m.[11] We add these counts for k = 180 measurement events, and then divide the resulting number by *L* and by *k*:

$$r = \frac{N}{k * L}$$

The result can be scaled to convenient units; for example "vehicles per km".

Note that this way of computing density averages the counts over a length of 37.5m, which is longer than most traffic detectors. The effect of this should be systematically studied.

- **Space Mean Speed, Travel Velocity**. It is well known that one can measure velocity either analogous to our flow definition (Time Mean Speed, Spot Speed) or analogous to our density definition (space mean speed, travel velocity). Under non-stationary conditions, the measurements give different results, since, for example, the first definition never counts vehicles with velocity zero. Time mean speed is easier for field measurements; space mean speed is easier to interpret since it is equal to the travel velocity and it is also the velocity that must be used in the fundamental relationship between flow, density, and velocity, $q = r \cdot v$. Since, in a simulation model, both are similarly easy to measure, we measure the more meaningful travel velocity. Once per timestep, we sum up the individual velocities of all vehicles on a stretch of roadway of *L*=5 sites =5 x 7.5m = 37.5m. We add these sums for *k*=180 measurement events and then divide the resulting number by *N* and by *k*, where *N* is the same number as obtained during the density measurement above:

$$v = \frac{\sum v}{k * N}$$

- **Lane usage**. Lane usage of a particular lane is the number of cars on this lane divided by the number of cars on all lanes. It can be computed as:

$$f_i = \frac{r_i}{\sum_{j=1}^{n} r_j \cdot n}$$

where *i* is the lane we look at, and *n* is the number of lanes.

[11] The "magical" number of L = 5 sites is equal to the maximum velocity of v_{max} = 5 sites / update. This ensures that each vehicle is counted at least once.

4.5.5.2 Test Networks

Essentially, two test networks are used: a circle of 1,000 sites = 0.75km in various configurations, and a simple signalized intersection. Most of the tests are run on the circle networks. The circle can have one, two, or three lanes. In all tests, the circle is slowly loaded with traffic via a parking location at site $x = 1$ (where the unit of x is *cells*). Velocity, flow, and density are measured on $486 \le x \le 490$, thus generating the fundamental diagrams for one-lane, two-lane, and three-lane traffic. Since the circle gets slowly loaded, the complete fundamental diagram is generated during one run.

For testing yield signs and stop signs, an incoming lane is added on the right side of traffic at $x = 501$. The characteristics of the incoming traffic are measured by a detector on the last five (5) sites of the incoming lane. The incoming lane is operated at maximum flow; i.e., with as many vehicles as possible entering. The incoming vehicles are removed at $x = 900$ via a parking accessory. The result of this measurement is typically a diagram showing the flow of incoming vehicles on the y-axis versus the flow on the circle on the x-axis.

For testing left turns against oncoming traffic, an opposing lane is added so that it ends at $x = 500$. The traffic control here is again a *yield* logic; the difference from before is that vehicles only *traverse* the opposing traffic, they do not join it.

Last, a three-lane intersection approach is used. The left lane makes a left turn, the middle lane goes straight, the right lane makes a right turn. Incoming vehicles have plans about their intended movement at the intersection and attempt to reach the corresponding lane. The intersection has signals with a one-minute green phase and a one-minute red phase. The typical output from this run is the flow of vehicles that go through the intersection, and the number of vehicles that cannot make their intended turn because they did not reach their lane.

4.5.5.3 Results

The results are shown in Figures 32 to 34.

- Single-lane traffic (Figure 32a) has a realistic value of maximum flow (= capacity), but one may argue that it is at a somewhat low density. The problem here is that we do not include slow vehicles; introducing slow vehicles into a single lane closed circle simulation just means that all faster vehicles bunch up behind them, which does not result in a very useful fundamental diagram. In terms of the *building block* philosophy, we prefer to run the single-lane test with identical vehicles.

  The flow through a traffic signal that is 50% green should be at half the value of the maximum single-lane flow; i.e., at 1000 veh/hour, which is what we find (Figure 32b).

- Our lane-changing rules do neither change maximum flow per lane nor the density (per lane) at maximum flow. That need not be the case [31]. Again, the density at maximum flow seems a bit low. This changes considerably when one introduces slower vehicles: The free flow part of the curve then bends more to the right, and the maximum is at higher densities [32]. Also, there are measurements in Germany where traffic *with* trucks reaches maximum flow at approx. 20—22 veh/km/lane [36], so without more specific data, this discussion seems pointless. We think that the curve without slow vehicles is *cleaner* and thus facilitates comparison between models; in reality, the problem is more complicated anyway.

Also, we generate equal lane usage between the lanes, as should be expected for a symmetric lane-changing model (in the absence of on-ramps).

- The curves for traffic through stop and yield signs follow the general form of the curve of the Highway Capacity Manual [3]. We added the HCM curves for comparison only. In general, we find that a yield sign, when there is no traffic on the major road, generates the same traffic as if there were no sign at all, which should be expected the way the simulation is set up. (It is a bit lower than for the *circle* before because the speed limit is lower here.) The stop sign generates a much lower flow in the same situation, because the explicit stop decreases capacity.

  From there, the curves for *traffic into* the major road decrease roughly linearly to zero when the flow on the major road reaches capacity. The curve for traffic across a single-lane road looks similar to its *traffic into* counterpart, which is to be expected because the number of opposing lanes is one in both cases. The curve for traffic across a two-lane road provides roughly half the flow of traffic across a single-lane road.

  For densities above capacity on the major road, all curves bend *back on themselves*. If the major road is congested, the speed there is zero, and the gap acceptance criterion "accept if $gap \geq 3 \cdot v_{onco\min g}$" is always fulfilled, even for *gap* = 0. Nevertheless, for *traffic into*, very little traffic makes it through the yield or the stop sign. The reason is that in TRANSIMS, vehicles on the major road that may go through the intersection *reserve* the first cell at the beginning of the next link, thus blocking this link for vehicles from the minor link even if the gap acceptance rule would allow the movement. For *traffic across*, this restriction does not exist, and many vehicles make it through the intersection, probably many more than is realistic. Note that the HCM does not provide information in the congested regime.

4.5.6 Yield Sign Behavior

All runs for this paper were first done with an experimental code and then repeated with the TRANSIMS production code; all results shown so far were obtained from the TRANSIMS production code. The disadvantage of an experimental code is that actual implementation in the production version may still introduce changes in the results due to small discrepancies.[12] The advantage of an experimental code is that turnover (compile times, complexity of code, etc.) is much better than with a production version. We used that advantage to test many different rules. In the following, we want to present a small subset of tests.

All results presented in this section refer to the situation of a one-lane minor street merging into a one-lane major street, with the intersection control being a yield sign. Figure 35 (a) shows what happens if the *reservation* rule from the TRANSIMS production code is no longer used. Clearly, if vehicles from the major road do reserve calls on the outgoing link only if they are actually going there, many more vehicles from the minor lane can make the turn, effectively leading to an *alternating* vehicle pattern. This may be desirable in some situations.

Figure 35 (b) shows what happens when one changes "accept when $gap \geq 3\ v_{oncoming}$" to "accept when $gap > 3\ v_{oncoming}$". This seems like a negligible difference in the rules; yet, the results are

[12] This explains the differences to the TRB preprint version of this paper, which contained results from the experimental code.

quite different in the congested regime. Whereas in the first, many vehicles are able to get into the congested major road, in the second, only a few of them make it. The difference is most easily explained by looking at a vehicle of speed zero on the major road just in front of the merge point, with space for a vehicle downstream of the merge point. With the first rule, a vehicle at the yield sign will accept the move and move in front of the vehicle on the major road; in the second case, it will not. Both scenarios seem to be plausible to us; only systematic measurements can probably resolve which one is better for a simulation model. Also note that the rule in (b) generates similar flows as the TRANSIMS production version.

Figures 35(b), 35(c), and 35(d) show the result of different speed limits (same speed limit for both streets). A high average free speed of approximately 130km/h (~80mph, generated by v_{max} =5), maybe a freeway merge, generates a flow of approximately 2000 veh/hour/lane in the incoming lane when there is no traffic on the major road (Figure 35 (c)). From there, maximum incoming flow decreases continuously. Lower average free speeds of approximately 75km/h (50mph, Figure 35 (b)) and 50km/h (30mph, Figure 35 (d)) generate lower maximum incoming flows and are generally closer to the Highway Capacity Manual curve. Yet, it should be clear that, contrary to the HCM, the *minor* flow is also a function of the speed limit and not only of the gap acceptance (the gap acceptance is the same in all three simulations).

A last series of experiments show the effect of different values for the gap acceptance. Figures 35(e) and 35(f) show "accept when $gap > v_{oncoming}$ and $gap > v_{max}$". Clearly, more vehicles are accepted, leading to a higher flow of turning vehicles as a function of the flow on the major road. Note that the flow via the yield sign is never higher than 1800 minus the flow on the major road. This reflects the fact that the major road cannot have a higher flow than 1800 veh/h/lane (free speed approximately 50mph); traffic through the yield sign can thus, at most, fill the major road to capacity. This explains why the acceptance of much smaller gaps does not produce a stronger difference. The situation is clearly different for unprotected turns *across* instead of *into* traffic, as can be seen for the left turns in the next section.

4.5.7 Comparison to Case Study Logic

The gap acceptance logic presented here and used in the March 1998 TRANSIMS microsimulation is different from the logic used in the *Dallas-Ft. Worth Case Study* [37, 38]. The logic during that case study was: "Accept an unprotected movement if in all opposing lanes the gap is larger than v_{max} =5." This means that at low density on the major road, more turns were accepted; whereas at high density on the major road, less turns were accepted – with the extreme case that no turns were possible against oncoming traffic of speed zero.

Figure 36 compares the results for the current gap-acceptance logic and the one used in the case study for the case where the major road is a three-lane road. Note that the results for the turns *into* other traffic are not that much different, whereas the result for the turns *across* other traffic yields much higher uncongested and much lower congested flows with the case study logic. This is due to the fact that for turns *into* other traffic, there is a capacity constraint of the form that the joint flows from the major and the incoming road cannot exceed capacity of the major road (see Section 4.5.9). Such a constraint obviously does not exist for turns *across* the major road.

4.5.8 Short Discussion

We presented a test of what we believe are *building blocks* of microsimulation models. Further *building blocks*, not included here, are probably freeway ramps with merge lanes, and freeway weaving sections. We plan to include these tests in future versions.

As pointed out earlier, we believe that *clean* real world measurements of the *building block* situations are hard to obtain. Thus, one may consider them primarily useful for comparing simulations with each other and with theory; nevertheless, we think that one can judge from the results at least if the simulation is *in the right ballpark*. It would certainly be desirable in the future to also have test suites for more complex situations. For the same reason, we did not make any attempt to get *better* results than the ones presented here: we know that the results change in more complex scenarios, and it is therefore unclear if a change *to the better* in the test cases may not be a change *to the worse* with respect to reality.

Also, we would shortly like to point out again that *verification* of simulation models (i.e., the question if an actual code corresponds to a (possibly incomplete) specification in a paper), is in practice a difficult question. An alternative approach would be to try to find a suite that decides if we are macroscopically convincing without the need to go through testing the rules on an individual scale. Arguing about the microscopic rules could then be left to a small group of specialists, the end user could just look at the test suite results and judge in a matter of minutes if the simulation has faults that would seriously affect the analysis of their problem.

Last, all of these problems imply to us that one should expect that simulation models will undergo continuous improvements, and it seems more realistic to us to expect *test suites* to be run at regular intervals instead of expecting that parts of simulation models can be validated and calibrated *once and for all* at certain stages and then never be touched again. In consequence, we would like to shift the argument from a discussion of whether a model is *correct or not* to the discussion about which tests should be run to enable the user to make that decision, and how these tests can be made comparable between different simulation models.

4.5.9 Summary and Conclusion

In transportation simulation models for larger scale questions such as planning, the flow characteristics of the traffic dynamics are, in some sense, more important than the microscopic driving dynamics of the vehicle itself. This becomes especially true since a *complete* representation of human driving is impossible anyway, because of both knowledge constraints computational constraints. Yet, calibrating a traffic simulation model against all types of desired behavior (for example against all HCM curves and values mentioned in this paper) seems to be a hopeless task given the high degrees of freedom.

TRANSIMS thus attempts to generate plausible emergent macroscopic behavior from *simplified* microscopic rules. This paper described the more important aspects of these rules as currently implemented or under implementation in TRANSIMS. Before we implement rules in the TRANSIMS production version, we usually try to run systematic studies with more experimental versions. The results of the traffic flow behavior from that study were presented. Also, we showed the effects of some changes in the rules for the example of a yield sign. Finally, some comparisons were made between the logic currently under implementation and the logic used for the Dallas-Ft. Worth case study.

One problem with microscopic approaches is that, in spite of all diligence, subtle differences between design and actual implementation can make a significant difference in the emergent outcome. For that reason, this paper should also be seen as an argument for a standardized traffic flow test suite for simulation models. We propose that simulation models, when used for studies, should first run these tests to demonstrate the dynamics of their emergent macroscopic flow behavior. We think that the combination of results presented in Figures 32, 33, and 34 are a good test set, although extensions may be necessary in the future (e.g., merge lanes, weaving, etc.). We will attempt to provide future TRANSIMS results also with updated versions of the results of the traffic flow tests.

4.5.10 References for *TRANSIMS Traffic Flow Characteristics*

[1] M. Van Aerde, personal communication.

[2] J. Van Leeuwen, editor. *Formal models and semantics*, volume B of *Handbook of Theoretical Computer Science*. Elsevier and MIT Press, 1990.

[3] Transportation Research Board. *Highway Capacity Manual*. Special Report No. 209, National Research Council, Washington, D.C., 3rd edition, 1994.

[4] M.J. Cassidy and J. Han. Validation and evaluation of freeway simulation models. Final report. Technical Report FHWA/CA/Purdue-RR-95-1, Purdue University, School of Civil Engineering, West Lafayette IN 47907, USA, 1995.

[5] H.S. Mahmassani, J.C. Williams, and R. Herman. Performance of urban traffic networks. In N.H. Gartner and N.H.M. Wilson, editors, *Transportation and Traffic Theory,* page 1. Elsevier Science Publishing Co., Inc., 1987.

[6] M. Ponzlet and P. Wagner. Validation of a CA-model for traffic simulation of the Northrhine-Westphalia motorway network. In *The 24th European Transport Forum, Proceedings*, volume P404-1, 1996.

[7] M. Cremer and M. Papageorgiou. Parameter identification for a traffic flow model. *Automatica*, 17(6):837-843, 1981.

[8] M. Cremer and H. Schütt. A comprehensive concept for simultaneous state observation, parameter estimation, and incident detection. In *Proceedings of the 11th Int. Symposium on Transportation and Traffic Theory,* Yokohama, Japan, 1990.

[9] R.W. Denney, J.C. Williams, S.C.S. Bhat, and S.A. Ardekani. Calibrating NETSIM for a CBD using the two fluid model. In *Large Urban Systems. Proceedings of the Advanced Traffic Management Conference*. Federal Highway Administration, 400 7th Street, SW, Washington, DC, USA, 1993.

[10] TRANSIMS, Transportation Analysis and SIMulation System, Los Alamos National Laboratory, Los Alamos, U.S.A. See www-transims.tsasa.lanl.gov.

[11] Michael Patriksson. *The Traffic Assignment Problem: Models and Methods.* Topics in Transportation. VSP, VSP, P.O. Box 346, 3700 AH Zeist, The Netherlands, 1994.

[12] R. Wiedemann. Simulation des Straßenverkehrsflusses. Schriftenreihe Heft 8, Institute for Transportation Science, University of Karlsruhe, Karlsruhe, Germany, 1994.

[13] M. Van Aerde, B. Hellinga, M. Baker, and H. Rakha. INTEGRATION: An overview of traffic simulation features. *Transportation Research Records*, in press.

[14] G. L. Chang, H.S. Mahmassani, and R. Herman. A macroparticle traffic simulation model to investigate peak-period commuter decision dynamics. *Transportation Research Record*, pages 107-120, 1985.

[15] T. Schwerdtfeger. *Makroskopisches Simulationsmodell für Schnellstraßennetze mit Berücksichtigung von Einzelfahrzeugen (DYNEMO).* PhD thesis, University of Karsruhe, 1987.

[16] R. Herman and I. Prigogine. A two-fluid approach to town traffic. *Science*, 204:148-151, 1979.

[17] K. Nagel. Freeway traffic, cellular automata, and some (self-organizing) criticality. In R.A. deGroot and J. Nadrchal, editors, *Physics Computing '92*, page 419, Singapore, 1993. World Scientific.

[18] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *J. Phys. I France*, 2:2221, 1992.

[19] C. L. Barrett, S. Eubank, K. Nagel, J. Riordan, and M. Wolinsky. Issues in the representation of traffic using multi-resolution cellular automata. Los Alamos Unclassified Report (LA-UR) 95-2658, Los Alamos National Laboratory, Los Alamos, NM, U.S.A., http://www.lanl.gov, 1995.

[20] A. D. May. *Traffic flow fundamentals.* Prentice Hall, Englewood Cliffs, NJ 1990.

[21] K. Nagel and M. Paczuski. Emergent traffic jams. *Phys. Rev. E*, 51:2909, 1995.

[22] S. Krauss, P. Wagner, and C. Gawron. Metastable states in a microscopic model of traffic. *Physical Review E*, 55(5):5597-5602, 1997.

[23] C.L. Barrett, M. Wolinsky, and M.W. Olesen. Emergent local control properties in particle hopping traffic simulations. In D.E. Wolf, M. Schreckenberg, and A. Bachem, editors, *Traffic and granular flow*, pages 169-173. World Scientific, Singapore, 1996.

[24] K. Nagel. Particle hopping models and traffic flow theory. *Phys. Rev. E*, 53(5):4655, 1996.

[25] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama. Dynamical model of traffic congestion and numerical simulation. *Phys. Rev. E*, 51(2):1035, 1995.

[26] R.D. Kühne and R. Beckschulte. Non-linearity stochastics of unstable traffic flow. In C.F. Daganzo, editor, *Proc. 12th Int. Symposium on Theory of Traffic Flow and Transportation*, page 367, Amsterdam, The Netherlands, 1993. Elsevier.

[27] B.S. Kerner and P. Konhäuser. Structure and Parameters of clusters in traffic flow, *Physical Review E*, 50(1):54, 1994.

[28] M. Rickert, K. Nagel, M. Schreckenberg, and A. Latour. Two lane traffic simulations using cellular automata. *Physica A*, 231:534, 1996.

[29] U. Sparmann. *Spurwechselvorgänge auf zweispurigen BAB-Richtungsfahrbahnen.* Number 263 in Forschung Straßenbau und Straßenverkehrstechnik. Bundesminister für Verkehr, Bonn-Bad Godesberg, Germany, 1978.

[30] P. G. Gipps. A model for the structure of lane-changing decisions. *Transportation Research B*, 20B(5):403-414, 1986.

[31] M. Rickert, K. Nagel, M. Schreckenberg, and A. Latour. Two lane traffic simulations using cellular automata. *Physica A*, 231:534, 1996.

[32] K. Nagel, D.E. Wolf, P. Wagner, and P. Simon. Two-lane traffic rules for cellular automata: A systematic approach, 1977. Submitted, also Los Alamos Unclassified Report (LA-UR) 97-4706, see www-transims.tsasa.lanl.gov/research_team/.

[33] P. Wagner, K. Nagel, and D.E. Wolf. Realistic multi-lane traffic rules for cellular automata. *Physica A*, 234:687, 1997.

[34] P. Wagner. Traffic simulations using cellular automata: Comparison with reality. In D.E. Wolf, M. Schreckenberg, and A. Bachem, editors, *Traffic and Granular Flow*, Singapore, 1996. World Scientific.

[35] C.L. Barrett. Personal communication.

[36] R. Wiedemann. Beschreibung des Staus. In H. Keller, editor, *Beiträge zur Theorie des Straßenverkehrs.* Forschungsgesellschaft für Straßen- und Verkehrswesen, Köln, Germany, 1995.

[37] R. J. Beckman et al. TRANSIMS-release 1.0 – the Dallas-Ft. Worth case study report. Los Alamos Unclassified Report (LA-UR) 97-4502, Los Alamos National Laboratory, see http://www-transims.tsasa.lanl.gov/research_team/, 1997.

[38] K. Nagel and C. L. Barrett. Using microsimulation feedback for trip adaptation for realistic traffic in Dallas. *International Journal of Modern Physics C*, 8(3):505-526, 1997.

4.5.11 Figures for *TRANSIMS Traffic Flow Characteristics*

FIGURES

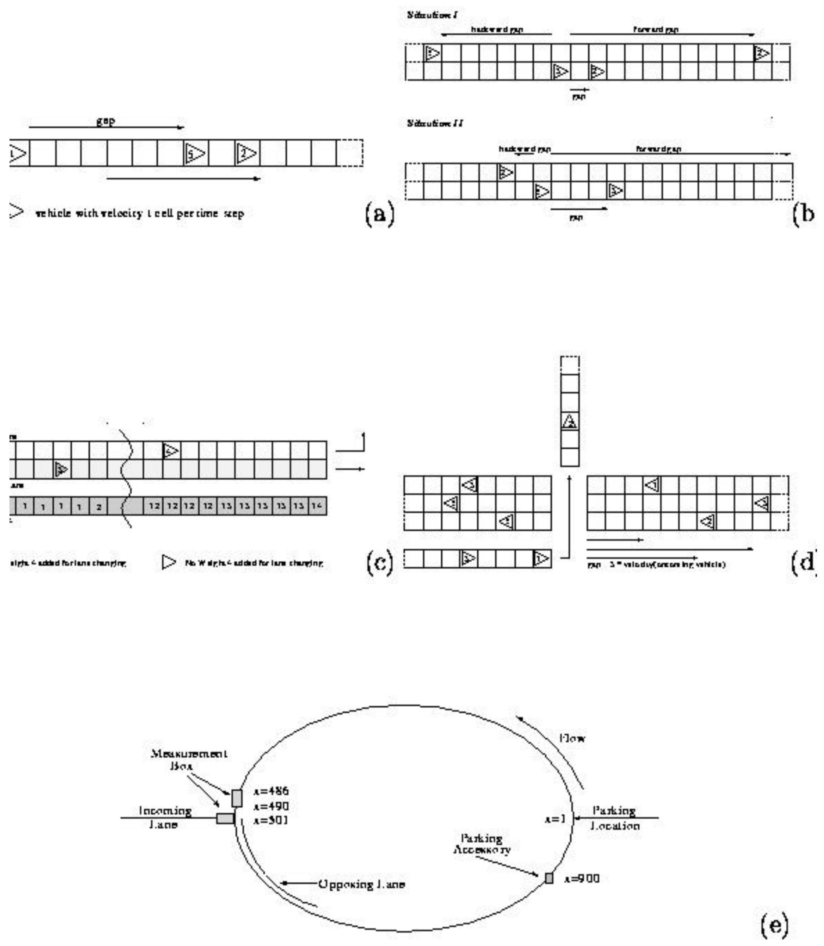


Figure 31

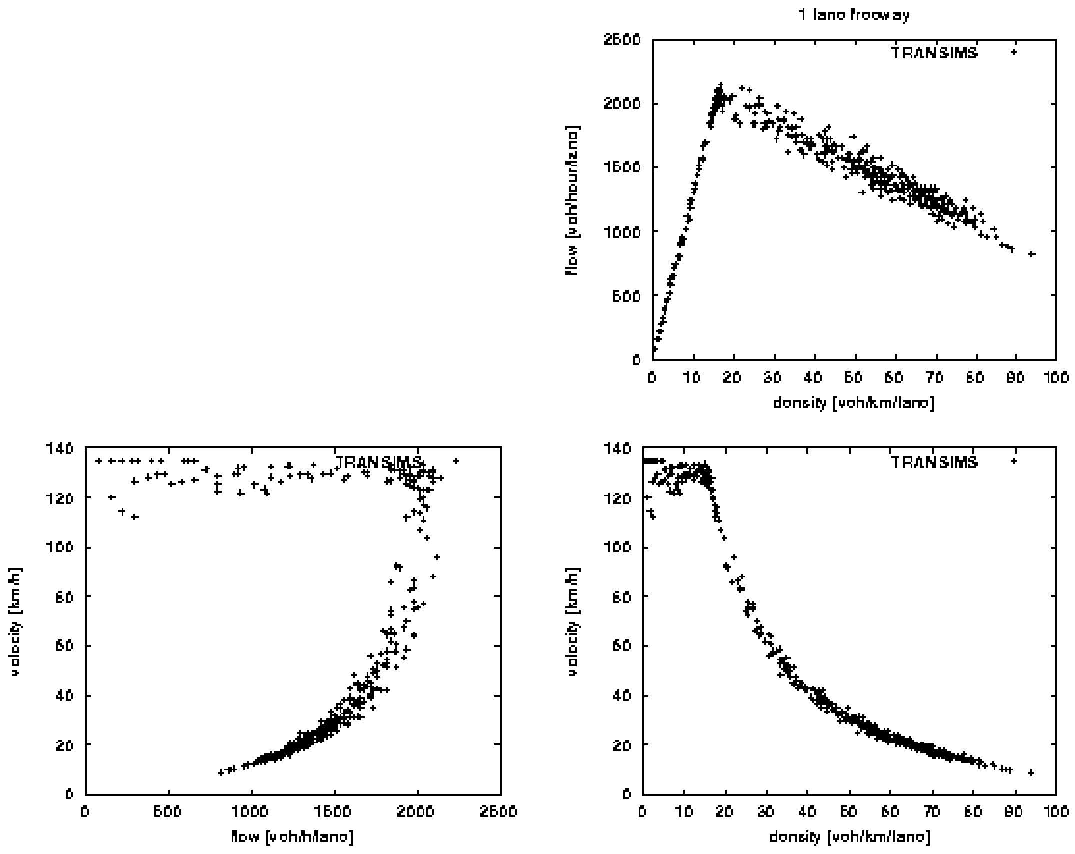
(a) Definition of *gap* and examples for one-lane update rules. Traffic is moving to the right. The leftmost vehicle accelerates to velocity 2 with probability 0.8 and stays at velocity 1 with probability 0.2. The middle vehicle slows down to velocity 1 with probability 0.8, and to velocity 0 with probability 0.2. The right-most vehicle accelerates to velocity 3 with probability 0.8 and stays at velocity 2 with probability 0.2. Velocities are in *cells per timestep*. All vehicles are moved according to their velocities at a later phase of the update.

(b) Illustration of lane-changing rules. Traffic is moving to the right; only lane changes to the left are considered. Situation I: The leftmost vehicle on the bottom lane will change to the left because (i) the forward gap on its own lane, 1, is smaller than its velocity, 3; (ii) the forward gap in the other lane, 10, is larger than the gap on its own lane, 1; (iii) the forward gap in the target lane is large enough: $weight2 = v - gap_o = 3 - 10 = -7 < 1 = weight1$; (iv) the backward gap is large enough: $weight3 = v_{max} - gap_b = 5 - 6 = -1 < 1 = weight1$. Situation II: The second vehicle from the right on the right lane will not accept a lane change because the gap backwards on the target lane is not sufficient.

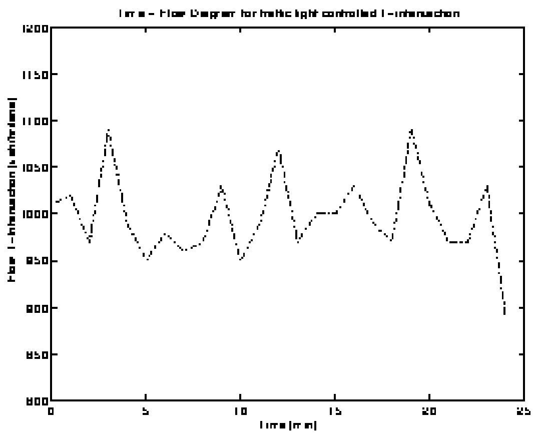
(c) Value of *weight4* when in the wrong lane during the approach to the intersection.

(d) Example of a left turn against oncoming traffic. The turn is accepted because on all three oncoming lanes, the gap is larger or equal to three times the first oncoming vehicle's velocity.

(e) Test networks.



(a)



(b)

Figure 32

(a) One-lane traffic: Flow vs. density, travel velocity vs. flow, and travel velocity vs. density.

(b) Number of vehicles going through the intersection and number of vehicles *off plan* (=0) per green phase, re-scaled to hourly flow rates per lane.

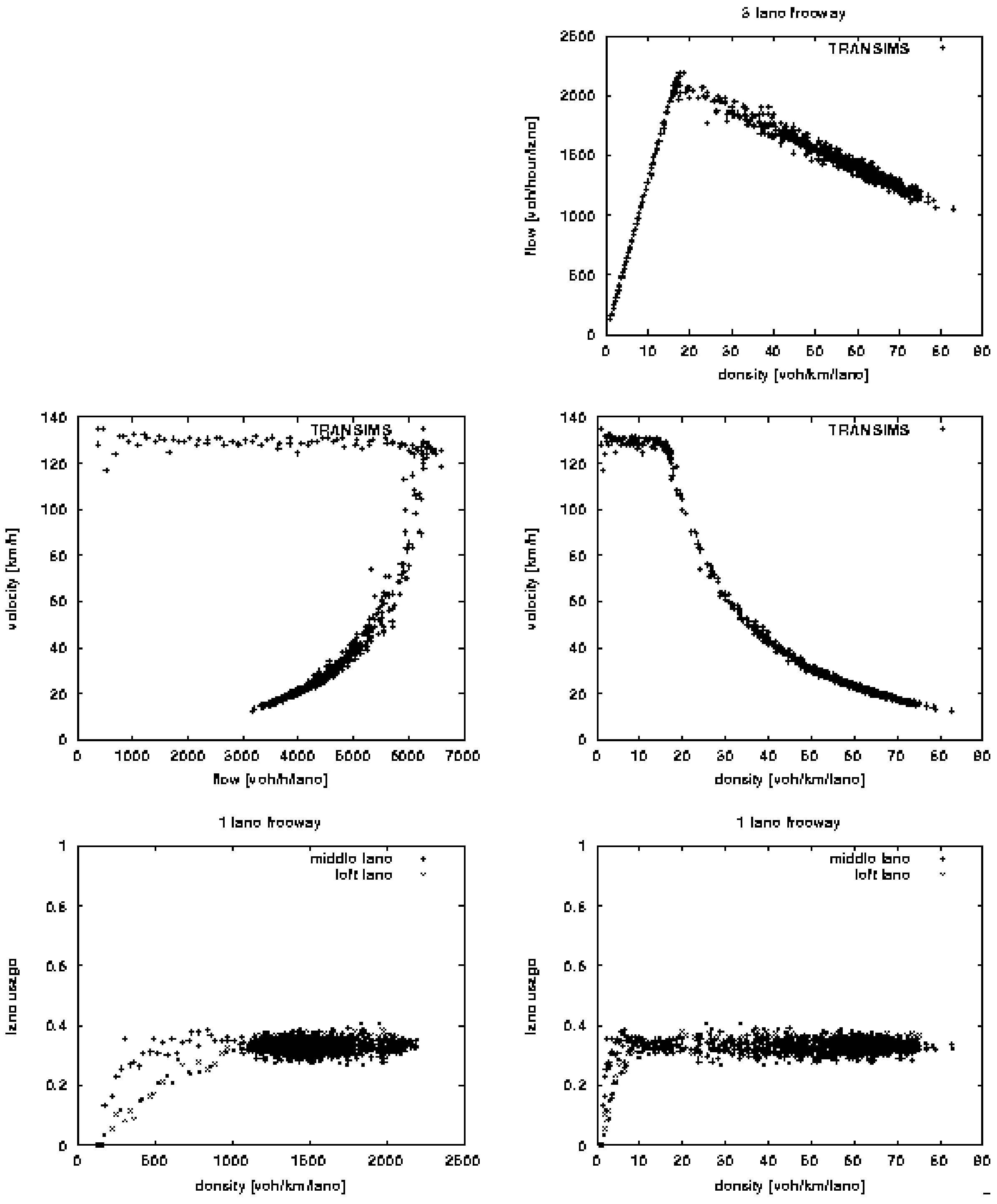


Figure 33

Three-lane circle: Flow vs. density, travel velocity vs. flow, travel velocity vs. density, lane usage vs. flow, and lane usage vs. density. The asymmetry in the lane usage at low densities is due to the fact that the parking locations start filling in vehicles on the right lane, and they only move to the left when traffic on the right lane becomes dense.

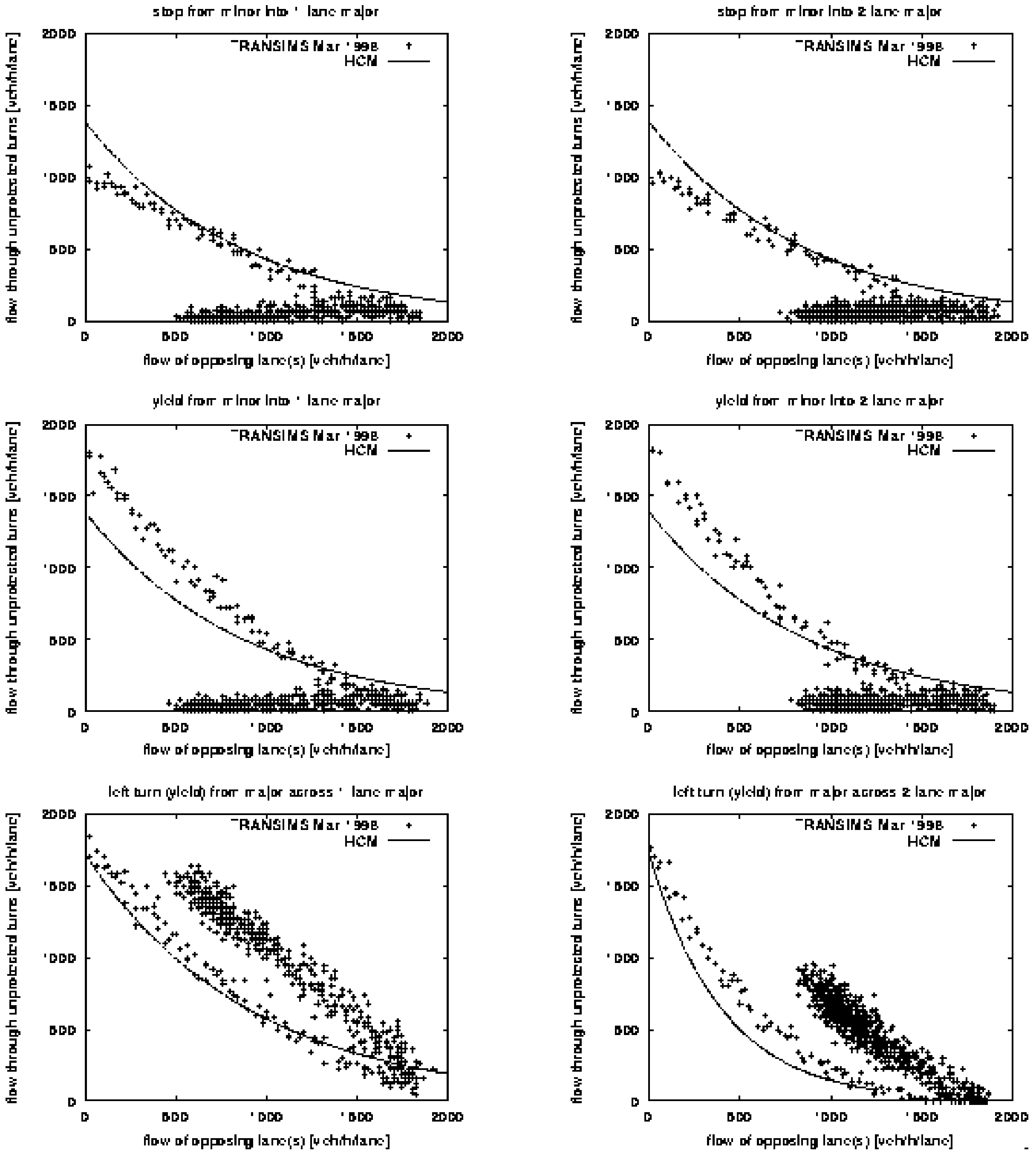


Figure 34

Flow through stop sign, yield sign, and unprotected left turn. Left column: Major road (*circle*) has one lane. Right column: Major road (*circle*) has two lanes. Solid line: Highway Capacity manual [3]. v_{max} - 3, gap acceptance rule is "accept if *gap* ³ $v_{oncoming}$, and if first site on target lane available". Note that for "left turn across two lanes" (bottom right), the opposing volume is the sum of both lanes, i.e., twice the value shown on the x-axis.

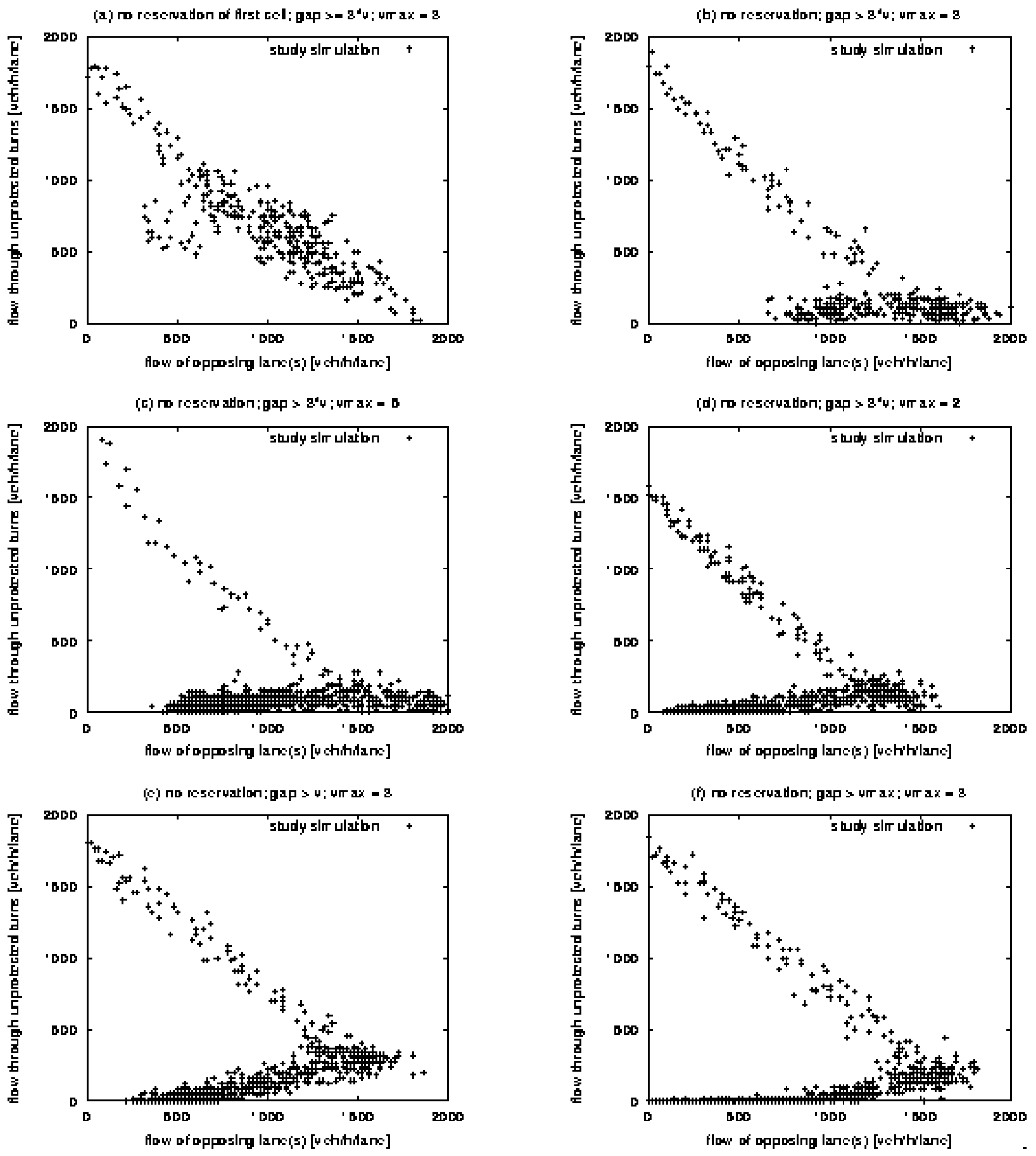


Figure 35

Comparison between different rules for the case of a one-lane minor road controlled by a yield sign merging into a one-lane major road.

(a) Same as Figure 34 (i.e., v_{max} = 3 and "accept if *gap* ≥3 $v_{oncoming}$"), except that traffic on major road does not reserve the first cell on the outgoing link, thus giving traffic from the yield sign more opportunities. Note that this seemingly small difference has big consequences in the congested regime.

(b) Same as (a) except that acceptance rule now "accept if *gap* > 3 $v_{oncoming}$".

(c) Same as (b) except that v_{max} = 5.

(d) Same as (b) except that v_{max} = 2.

(e) Same as (b) except that acceptance rule now "accept if *gap* > $v_{oncoming}$".

(f) Same as (b) except that acceptance rule now "accept if *gap* > v_{max}".

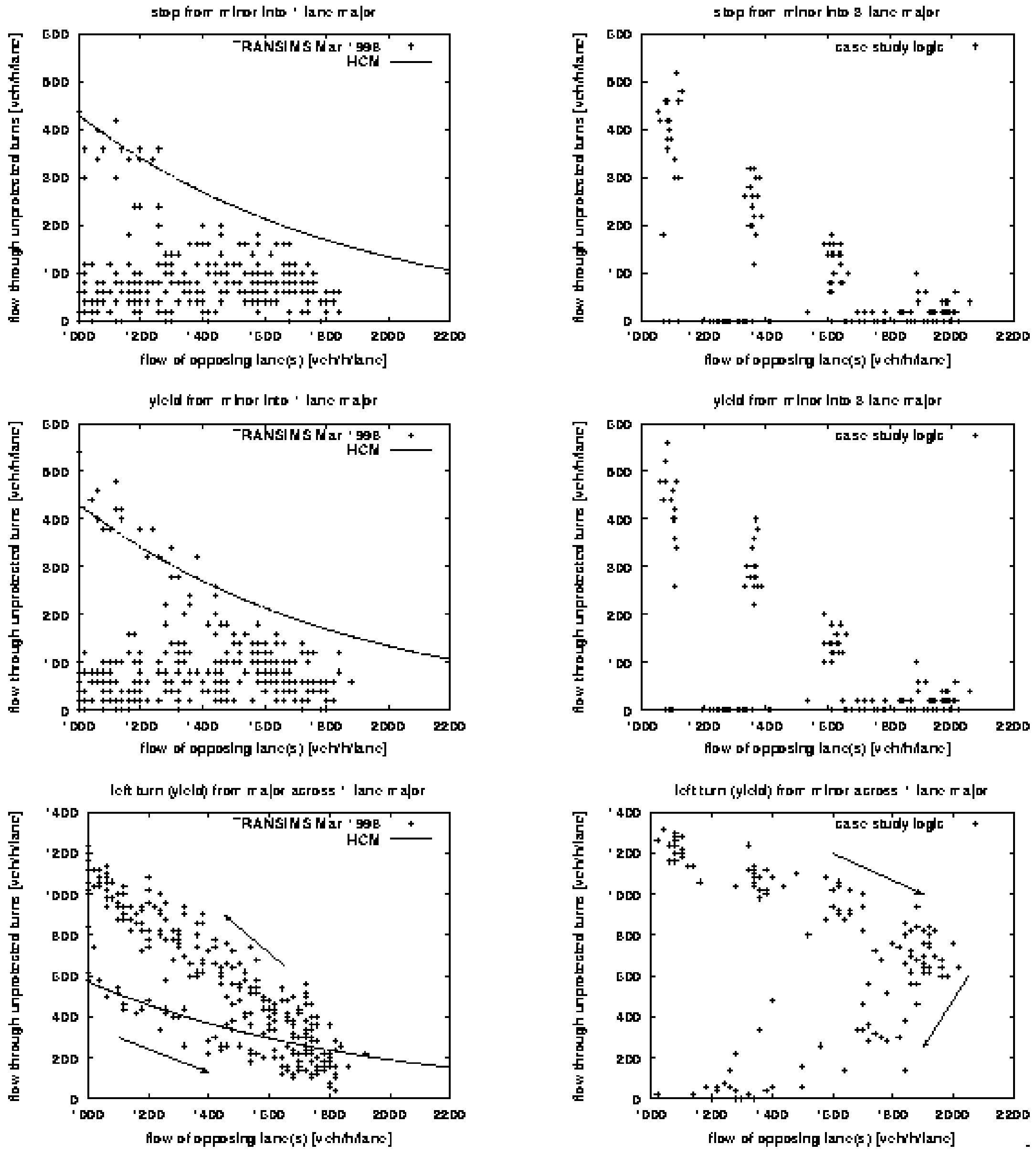


Figure 36

Comparison between the March 1998 TRANSIMS microsimulation gap acceptance logic and the one used in the case study. Flow through stop sign, yield sign, and unprotected left turn into/across major traffic on major road. Left column: March 1998 TRANSIMS microsimulation. Right column: case study TRANSIMS microsimulation. The arrows in the left turn case indicate the direction of increasing congestion. The results are not strictly comparable because (i) the simulations in the right column were run with a maximum speed of v_{max} = 5 cells/update (135 km/h) vs. v_{max} - 3 cells/update (81 km/h) in the left column (mostly noticeable in the lower maximum flow on the major road); and (ii) the stop and yield cases on the right describe flow into a three-lane road vs. flow into a one-lane road in the left column. Note that the results for the turns *into* other traffic (*stop* and *yield*) are not that much different between the two, whereas the result for the turns *across* other traffic (*left turn*) leads to much higher flows in the uncongested and lower flow in the congested regime with the case study logic.

5. TROUBLESHOOTING

It is our intention to shield the user from as many problems as is feasible. For example, the TAI checks to make sure that all information is supplied and that data is in the correct range before starting a simulation. This section gives scenarios of problems that may still occur and, where possible, gives corrective action. Problems whose solutions are not obvious have been included. Problems that have easy-to-understand error messages have been excluded.

5.1 Running TAI

Problem 5.1.1

The interface is displayed on the screen in black and white with a wide black horizontal strip running across the middle.

Solution 5.1.1

There is important information under the black strip; this problem should be fixed before you continue. Other applications such as Netscape or ArcView are running that have depleted the color table. Exit the TAI and the other applications. Restart the TAI. If the TAI is still displayed in black and white, exit your window server. Restart the window server and the TAI.

Experimentation will show you which applications deplete the color table. The applications can be started without affecting the appearance of a currently running TAI.

Problem 5.1.2

When the TAI is started, information must be supplied that is normally supplied during logon.

Solution 5.1.2

There are queries in your .cshrc file. Look through your .cshrc file and any files sourced by your .cshrc file for lines that have “$<” in them, such as

```
set response = $<
```

These queries must be taken out or moved from your .cshrc file to your .login file.

Problem 5.1.3

The font size is too small to be read easily in the output log.

Solution 5.1.3

Messages in output log windows can be highlighted and copied into a text editor for easier viewing, scrolling, and searching.

Problem 5.1.4

The output in the Simulation Output window is truncated.

Solution 5.1.4

The Simulation Output log is a partial copy of the file mentioned in the header bar of the window. You can open that file in your text editor. The file has not been truncated.

Problem 5.1.5

The View Log button is enabled even though a simulation has not yet been run.

Solution 5.1.5

The TAI keeps track of the last simulation that was started. The View Log button refers to the log associated with that simulation. The job status information, such as the "Microsimulation Running" status also refers to that job.

Problem 5.1.6

The "Microsimulation Running" status says Yes, even though a simulation has not yet been run.

Solution 5.1.6

The TAI keeps track of the last simulation that was started. The job status information refers to that job. You may have exited and restarted the TAI since you started the simulation. If so, the simulation is probably still running. Click **[View Log]** to check on the progress of that job.

Another possibility is that the simulation may have been killed by a machine crash or some other cause that prevented a graceful exit. This leaves a status file on the system that is incorrect. Click **[View Log]** to check on progress of the job. If it is clear to you that the simulation has already been killed, click **[Stop Microsim]**. This will set the running status to **No** and will enable you to run another simulation.

Problem 5.1.7

After starting the TAI, the following message is displayed:

```
The environmental variable TRANSIMS_HOME is not set.

The following lines must be included in the .cshrc file in your home
directory, after PATH, LD_LIBRARY_PATH, and MANPATH have been set.

  setenv TRANSIMS_HOME your_transims_home
  source $TRANSIMS_HOME/data/transims_home

For your_transims_home, substitute the directory where TRANSIMS'
software is installed.
/opt/transims is the default installation directory.
```

After you have added these lines, run `source ~/.cshrc`

Solution 5.1.7

If you get an error message similar to this and feel that you have followed the included instructions, check for the following problems:

Make sure that the following two lines are in your .cshrc file, in this order:

```
  setenv TRANSIMS_HOME your_transims_home
  source $TRANSIMS_HOME/data/transims_cshrc
```

It is not enough to just run these commands from a shell. Run the following commands:

```
  % source ~/.cshrc
  % printenv TRANSIMS_HOME
```

It is possible that there is an *exit*, *if*, *while*, or *for* command in your .cshrc file that is preventing the TRANSIMS_HOME lines from being run.

Problem 5.1.8

After starting the TAI, the following message is displayed:

```
envcheck: unsetting TRANSIMS_HOME and sourcing ~/.cshrc
```

and nothing else.

Solution 5.1.8

The problem is in your .cshrc file. It is probably waiting for input from the user. An example of this would be a prompt for your terminal type, after which you would type `vt100` or `xterm`. Any code in your .cshrc file that asks for input before continuing will not work with TAI. This code should be in your .login file instead. Kill the job that is running. Move the code that asks for input from your .cshrc file to your .login file.

To test, run

```
  % source ~/.cshrc
```

It should run to completion without needing any input from the keyboard.

Problem 5.1.9

After starting the TAI in background, the following message is displayed:

```
envcheck: unsetting TRANSIMS_HOME and sourcing ~/.cshrc
2]  + Stopped (tty output) tai
```

and nothing else.

When the TAI is run in the foreground, it runs fine.

Solution 5.1.9

The problem is in your .cshrc file. Some commands, such as stty, ask for input from the keyboard when called from a job that is running in background. But they do not need input when they are called from a job running in foreground. Move any stty commands in your .cshrc file to your .login file.

5.2 Running TAI – Unworkable DISPLAY

Problem 5.2.1

After starting the TAI, the following message is displayed:

```
The DISPLAY is set to yourMachine
TRANSIMS_HOME=/opt/transims
Using default microsim output root directory /opt/transims/output
java.lang.InternalError: Cannot connect to X11 window server using
'yourMachine' as the value of the DISPLAY variable.
   at sun.awt.motif.MToolkit.<init>(MToolkit.java:48)
   at java.awt.Toolkit.getDefault Toolkit (Toolkit.java:402)
   at java.awt.Window.getToolkit (Window. java: 222)
   at java.awt.Frame.addNotify (Frame.java:201)
   at Frame1.<init>(Frame1.java: 366)
   at Frame1.main(Frame1.java: 1788)
```

Solution 5.2.1

You have not set the environment variable DISPLAY correctly. In the example, you (or your .login file or .cshrc file) have set DISPLAY to *yourMachine*. For the purposes of this example, assume that *yourMachine* is the name of the machine on which your window manager is running. The problem is that you have not appended the string ":0.0" on the end of *yourMachine*.

First, kill the job that produced the error message.

Then, run the following command:

```
   % setenv DISPLAY yourMachine:0.0
```

substituting the name of the machine on which your window manager is running for *yourMachine*.

Finally, rerun the TAI.

Problem 5.2.2

After starting the TAI, the following message is displayed:

```
Your DISPLAY is set to bogusMachine:0.0
TRANSIMS_HOME=/opt/transims
Using default microsim output root directory /opt/transims/output
java.lang.InternalError:  Cannot connect to X11 window server using
'bogusMachine:0.0' as the value of the DISPLAY variable.
   at sun.awt.motif.MToolkit. <init>(MToolkit.java:48)
   at java.awt.Toolkit.getDefault Toolkit(Toolkit.java:402)
   at java.awt.Window.getToolkit(Window.java:222)
   at java.awt.Frame.addNotify(Frame.java:201)
   at Frame1.<init>(Frame1.java:366)
   at Frame1.main(Frame1.java:1788)
```

Solution 5.2.2

You have set the environment variable DISPLAY to a machine that is not on your network. In the example, you (or your .login file or .cshrc file) have set DISPLAY to *bogusMachine:0.0*.

First kill the job that produced the error message.

Then, run the following command:

```
   % setenv DISPLAY yourMachine:0.0
```

substituting the name of the machine on which your window manager is running for *yourMachine*.

Finally, rerun the TAI.

Problem 5.2.3

After starting the TAI , the following message is displayed:

```
Your DISPLAY is set to yourMachine:0.0
TRANSIMS_HOME=/opt/transims
Using default microsim output root directory /opt/transims/output
   Xlib: connection to "yourMachine:0.0" refused by server
   Xlib: Client is not authorized to connect to Server
java.lang.InternalError: Cannot connect to X11 window server using
'yourMachine:0.0' as the value of the DISPLAY variable.
   at sun.awt.motif.MToolkit.<init>(MToolkit.java:48)
   at java.awt.Toolkit.getDefaultToolkit(Toolkit.java:402)
   at java.awt.Window.getToolkit(Window.java:222)
   at java.awt.Frame.addNotify(Frame.java:201)
   at Frame1.<init>(Frame1.java:366)
   at Frame1.main(Frame1.java:1788)
```

Solution 5.2.3

For the purposes of this example, assume that *yourMachine* is the name of the machine on which your window manager is running. You have correctly set the environment variable DISPLAY to yourMachine:0.0. The problem is that you must run the xhost command on *yourMachine*. Note the "Xlib" lines in the error message.

Open a shelltool or cmdtool on your machine.

Run the command

```
% xhost taiMachine
```

where *`taiMachine`* is the name of the machine where you are running the "tai" command to start TAI.

Then, make the window logged on to *`taiMachine`* the active window again.

Kill the job that produced the error message.

Finally, rerun the TAI.

If you continue to get the error message, perhaps you need to fully qualify *taiMachine*. For example, we have a machine named gershwin.tsasa.lanl.gov. The commands

```
xhost gershwin.tsasa
```

and

```
xhost gershwin.tsasa.lanl.gov
```

work. The command

```
xhost gershwin
```

does not work.

Sometimes it helps to try running a program that you know works on your system, such as xterm. If the xterm window appears, you know that DISPLAY is set correctly and that xhost has been run correctly.

5.3 Entering Data in the Microsimulation Setup and Driver Logic Windows

Observe the shelltool or cmdtool window from which you started the TAI. If you leave a data field empty in the setup window, error messages will be written to the shell when you click **[OK]**. The data associated with the empty field will revert to its original value. The field itself will remain empty until you fill in a value.

Many data entry errors are not reported until the simulation is started via the Run or Run Calib buttons. Examples of this are out of range values or non-existent files. When you click **[Run]**, the problem will be reported in the Simulation Output window. The simulation will not run until you have fixed all of your input errors.

Problem 5.3.1

The Simulation End Time was changed from 10:00 to 6:30, but the simulation ran until 10:00. (This is just one example. The same problem could be found with any of the TAI's time fields or many of the other data fields.)

Solution 5.3.1

There are two possible causes for the problem.

1) You did not click **[OK]** in the Microsimulation Setup window before running the microsimulation. Click **[OK]**.
2) You have an empty field in the hour, minute, or seconds box of the Simulation End Time. Open the Microsimulation Setup window. Fill in the three Simulation End Time fields. Click **[OK]**. Click **[Run]** to start the simulation.

Problem 5.3.2

Some of the text in the Microsimulation Setup window is off the edge of the window.

Solution 5.3.2

Resize the window a tiny bit. The window will then redraw itself correctly.

Problem 5.3.3

There is a machine on the network that a user wants to use as a master or a slave, but it does not appear in the list of machines in the Microsimulation Setup window.

Solution 5.3.3

Toggle between the Multiprocessor and Workstation/LAN selections. Look to see if the desired machine is on either list. If it is not, your system manager can add a reference to it in the file $TRANSIMS_HOME/data/TAI/machines.lis. See the Installation Instructions.

5.4 Running the Microsimulation

When you click **[Run]**, the TAI makes a cursory check of your data to make sure that all of the necessary data has been supplied. It checks for the existence of files that need to be read and for the writability of files that need to be written. Where possible, values that you have specified are checked for correct ranges. If any of these tests fail, you will see a Data Errors window, and the simulation will not be run. Correct the input and rerun.

If your input data is complete and passes the preliminary checks, the simulation script named *runsim* starts running. Its output appears in a window that says "Simulation output in ..." at the top, where ... is the name of the log file that is being shown. The runsim script runs other programs including WriteOutputSpecs, PlanPreProcess, runCA, CA, ProcessCAOutput, runpvm, PVM, runhalt, and killpvm. If any of these programs fails, runsim stops and reports a non-zero error status. In that case, read the end of the log carefully to see if you can figure out what went wrong.

Usually, the problem will be obvious. There are too many possible errors to list each one here. The following section shows some problems that have solutions that are not so obvious.

Problem 5.4.1

An error message was displayed in the Simulation Output window, but it was impossible to scroll back to it because the cursor jumps to the end of the text.

Solution 5.4.1

In the Simulation Output window, click the mouse button anywhere other than the end of the text. The cursor will stop jumping to the end of the text as new text is added. You will now be able to scroll through the file.

Problem 5.4.2

How does one search for a string, such as ERROR, in an output log?

Solution 5.4.2

Messages in output log windows can be highlighted and copied into a text editor. You can then use the text editor's search capability. The simulation output log is a copy of the file mentioned in the header bar of the log's window. Open that file in your input editor.

Problem 5.4.3

The simulation terminates with the following message:

ERROR: unable to connect to database . . . Exiting.

Solution 5.4.3

You have not supplied the correct location of the database. In the TAI window, click **[Microsim Setup]**, then click **[Other]** at the top of the Microsimulation Setup window. The "Database name" field should have the full path to your database. If you list the files in this directory, you should see many files ending with the suffixes .lck and .tbl. If not, get the correct "Database name" path from the person who installed TRANSIMS.

The "Database Username" and "Database Password" fields are currently ignored by the database system.

Problem 5.4.4

When one clicks **[Run]** to start a second microsimulation, the following message is displayed.

Microsimulation already running. Use Stop Microsim button to terminate previous run.

Solution 5.4.4

Each individual user is limited to running one simulation at a time. The workaround is for the system manager to provide the user with more than one user id.

Problem 5.4.5

When running a simulation, the message *Permission denied* is displayed in the Simulation Output window, and the simulation exits.

Solution 5.4.5

TRANSIMS has not been configured correctly. Either the master or one of the slave machines you selected is not in the file $TRANSIMS_HOME/data/TAI/rhosts. Your system manager must alter that file. See the Installation Instructions.

5.5 Microsimulation Output Viewer

Problem 5.5.1

When you click **[Step]** in the Microsimulation Output Viewer window, nothing happens. The message *Unable to Step... Data Not Available* is displayed below the map.

Solution 5.5.1

There are many possible solutions to this problem.

1) No vehicles in the simulation. There were no vehicles in the simulation during the specified time interval on the links and nodes you specified. This is often the case if you specified a very short simulation or output collection time span. Or, you may have specified output collection over only one or two lightly traveled links.

2) Vehicle data file does not exist. Check the Microsimulation Output Viewer Log window for error messages. If you chose a Display Mode of Snapshot, look for the following message:
   WARNING: Unable to open Vehicle Data File --> Not drawing vehicles
   If you chose a Display Mode of Box Density, look for the following message:
   WARNING: Unable to open Summary Box Data File
   This indicates that the Vehicle Data File specified in the Microsimulation Output Viewer window does not exist. You can verify this by running ls on that file. Something may have gone wrong in the simulation that prevented the Vehicle Data File from being written. Look at the simulation's log file for any problems. One possibility is that something failed in postprocessing. Look in the log for the string "ProcessCAOutput", which is the name of the postprocessing program. The Vehicle Data File and other files needed by the Microsimulation Output Viewer are written by ProcessCAOutput.

3) Bad values in the Microsimulation Output Viewer window. Values in the Microsimulation Output Viewer window must agree with those in the Microsimulation Setup window *at the time that the simulation was started*. For this reason, it is a good idea to click **[Save]** just before you start a simulation, so that you can open the trnfile later to check your Output Viewer values. It is not difficult to figure out which data in the Setup window corresponds with a given piece of data in the Output Viewer window. The one data name that may be confusing is the data for Summary Sampling Interval. This corresponds to the Summary Data Box Sampling Interval in the Setup window.

Problem 5.5.2

When the value of the Box Length in the Microsimulation Setup window was changed, the simulation ran OK, but the results in the Output Viewer were not as expected.

Solution 5.5.2

The Network metafile specified in the Microsimulation Output Viewer window is tied to a given box length. At present, all Output Viewer networks are set up for a box length of 150 meters.

5.6 Running the Plan Viewer

Problem 5.6.1

Switching between the Plan Viewer window and other windows causes the colors in all windows to change.

Solution 5.6.1

When the mouse is in the Plan Viewer window, press **<Ctrl-Again>** or **<Ctrl-L2>**, depending on your keyboard. If you want the colors in another window to change, move the cursor to that window and press **<Ctrl-Again>** once more.

Warning: once you do this, you will have to continue to press the key sequences between windows until you exit your window manager.

5.7 Running ArcView

ArcView is started when one clicks **[Start ArcView to create link/node file]** in the Microsimulation Setup window. It is also started one clicks **[Network]** in the TAI window.

Problem 5.7.1

[Start ArcView to create links/nodes file] has been clicked. When ArcView comes up, the "Save Highlights to File(s)" or "Display Highlights from File(s)" menu items are not visible.

Solution 5.7.1

You do not have a file named default.apr in your home directory.

5.8 Editing Long Filenames in Textboxes

Problem 5.8.1

The filename in a textbox is so long that it is wider than the textbox. Is it possible to scroll to the text that is outside the box?

Solution 5.8.1

Click the mouse inside the textbox. Hold the mouse button down while you move the cursor out of the textbox to the right or left. The text will scroll in that direction.